Računarska grafika

Beleške za predavanja

Vesna Marinković
email: vesna.marinkovic@matf.bg.ac.rs
URL: www.matf.bg.ac.rs/~vesna.marinkovic
Matematički fakultet, Beograd

©2023

Autor:
dr Vesna Marinković, docent Matematičkog fakulteta u Beogradu

RAČUNARSKA GRAFIKA

Sva prava zadržana. Nijedan deo ovog materijala ne može biti reprodukovan niti smešten u sistem za pretraživanje ili transmitovanje u bilo kom obliku, elektronski, mehanički, fotokopiranjem, smanjenjem ili na drugi način, bez prethodne pismene dozvole autora.

Predgovor

Ovaj tekst predstavlja prateći materijal za kurs "Računarska grafika" na Matematičkom fakultetu Univerziteta u Beogradu. Nastao je na osnovu skripte prof. dr Predraga Janičića koji je ovaj kurs držao tokom od 2002/03 do 2004/05 i od 2007/08 do 2013/14. Tekst je prvenstveno zasnovan na narednim materijalima:

- John F. Hughes et al: *Computer Graphics: principles and practice* (3rd. ed),
- Peter Shirley et al, *Fundamentals of Computer Graphics* (3rd ed.),
- predavanja i slajdova za kurs ,,Računarska grafika" Univerziteta Carnegie Mellone
- slajdovi za kurs ,,Uvod u računarsku grafiku" Univerziteta Brown

ali i na mnogim drugim izvorima.

Materijal je namenjen studentima da im olakša praćenje nastave i pripremu ispita, ali ni na koji način ne može zameniti pohađanje nastave.

Veliku pomoć u izradi trenutne verzije skripte dobila sam od prof. dr Predraga Janičića i asistena Marka Spasića. Zahvaljujem se takođe i studentima na mnogobrojnim korisnim komentarima, posebno studentima Lazaru Vasoviću, Lazaru Perišiću, Gorani Vučić i Jeleni Simović. S obzirom na to da je materijal i dalje u fazi dorade, molim sve pažljive čitaoce da ako uoče neku grešku, jave autoru skripte.

Vesna Marinković,
Beograd, oktobar 2023.

Sadržaj

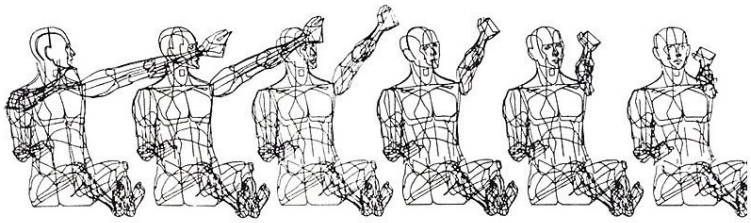
Glava 1

Uvod

1.1 Pojam računarske grafike

> "Jedna slika vredi hiljadu reči"
> kineska poslovica

Računarska grafika (eng. computer graphics) se može definisati kao tehnologija koja omogućava generisanje, prikaz i manipulaciju vizualnog sadržaja na ekranima računara. Razvila se sredinom dvadesetog veka sa idejom da olakša komunikaciju sa raŭnarima. Termin računarska grafika vodi poreklo od Viljema Fetera (William Fetter), američkog grafičkog dizajnera koji se smatra pionirom u oblasti računarske grafike. On je šezdesetih godina prošlog veka uveo termin računarska grafika kako bi opisao nove metode dizajniranja koje je koristio prilikom razvoja ergonomskih rešenja za kokpit Boing aviona. Naime, on je tokom rada na dizajnu kokpita, uz pomoć računara napravio niz slika modela pilota Boing aviona kako bi se testirala mogućnost da dohvati sve instrumente (slika 1.1). Ovo je bio prvi računarski generisan trodimenzioni model ljudskog tela. Na ovim osnovama se ubrzo nakon toga razvio dizajn uz pomoć računara (eng. Computer Aided Design, skraćeno CAD).



Slika 1.1: Prvi računarski generisani model ljudskog tela, tzv. "Boeing man" (izvor: Boeing Image Archive).

Računarska grafika se u užem smislu bavi modelovanjem objekata na sceni, modelovanjem osvetljenja scene i – na osnovu dobijenih modela – iscrtavanjem scene iz tačke gledišta kamere. Modelovanje objekata obuhvata specificiranje geometrijskog opisa objekata na sceni i načina na koji objekti reflektuju svetlost, dok modelovanje osvetljenja podrazumeva zadavanje matematičkog opisa izvora svetlosne energije, smerova u kojima se ona emituje, raspodele talasnih dužina svetlosti i dr. Rad u oblasti računarske grafike podrazumeva stvaranje, skladištenje i rukovanje modelima i slikama, pri čemu modeli mogu poticati iz različitih domena, poput fizike, biologije, matematike i umetnosti i mogu da prikazuju stvarne objekte, uključujući objekte koji se ne mogu videti u stvarnom svetu (na primer, atome) ili zamišljene objekte. Dobijene slike mogu biti u potpunosti sintetičke ili dobijene obradom fotografija.

Uz fotografiju i televiziju, računarska grafika danas predstavlja najdominantniji način proizvodnje slika. Prednost računarske grafike je mogućnost prikazivanja apstraktnih objekata, poput podataka koji nemaju ugrađenu geometriju, kao što su recimo numerički rezultati nekog eksperimenta.

Računarsku grafiku karakteriše interdisciplinarnost: u njoj važnu ulogu igraju fizika (potrebna za modelovanje svetlosti i izvođenje simulacija potrebnih za animacije), matematika (potrebna za opis ge-

ometrije figura), čovekova moć opažanja (bitna za alokaciju potrebnih resursa jer nije potrebno trošiti vreme za obradu objekata koji neće biti vidljivi ili na prikaz detalja koje čovekov vizualni sistem neće primetiti), interakcija između čoveka i računara, grafički dizajn, umetnost i druge. Sve ove aspekte je neophodno temeljno razmotriti kako bi dobijena slika bila što realističnija.

Oblast računarske grafike je matematički zasnovana i teorijski dobro ustanovljena. Hardverska i softverska rešenja specijalizovana za računarsku grafiku postoje decenijama i brzo se razvijaju i unapređuju i zbog toga ćemo se u ovom materijalu baviti konceptima koji su zajednički za sve implementacije, bez ulaženja u detalje nekog konkretnog hardvera ili neke softverske biblioteke.

1.2 Primene računarske grafike

Računarska grafika je od svojih početaka pa do danas iz noviteta prerasla u svakodnevni fenomen. Kako je grafika sve više postajala deo čovekove svakodnevice, tako su rasla i očekivanja korisnika. Video igre danas prikazuju više milijardi poligona u sekundi, a specijalni efekti su toliko dobri da ih nije lako razaznati od materijala koji nije računarski generisan. Digitalni fotoaparati i kamere prave slike visoke rezolucije, a alati za njihovu obradu veoma brzo napreduju. U isto vreme sve veća snaga računara omogućila je i bogatiju formu grafike.

Možda je najznačajniji uticaj računarske grafike u oblasti grafičkih korisničkih interfejsa (eng. graphical user interface, GUI) na različitim uređajima kao što su telefoni i računari, ali i kontrolne table automobila i brojni drugi kućni elektronski uređaji. Grafički korisnički interfejsi su omogućili prirodniju i lakšu interakciju sa računarom korišćenjem paradigme pokaži-i-klikni (eng. point and click).

Računarska grafika je promenila način na koji posmatramo informacije iz oblasti medicine, arhitekture, kontrole procesa u industriji, kao i informacije iz svakodnevnog života. Tako, na primer, možemo da vizualizujemo mape vremenske prognoze i radarske slike. Vizualizacija informacija je veoma korisna za razumevanje i interpretiranje podataka velikog obima i kompleksnih podataka koji nemaju nužno prirodni vizualni prikaz. Vremenski trend cena različitih akcija može se vizualizovati pametnim algoritmima za izradu grafikona tako da ljudi mogu da uoče obrasce u ovim podacima, iako oni na prvi pogled nisu vidljivi. Vizualizacija podataka u medicini omogućava pravljenje smislenih slika skeniranih pacijenata: na osnovu velike serije dvodimenzionih rendgenskih snimaka dobijenih kompjuterskom tomografijom (CT), mogu se kreirati osenčene slike unutrašnjosti snimljenog objekta, koje pomažu doktorima da iz ovakvih podataka izdvoje glavne informacije.

Računarska grafika se koristi i za dizajniranje delova i proizvoda na računarima (eng. Computer Aided Design, skraćeno CAD), a zatim, korišćenjem ovih virtuelnih dizajna, za vođenje procesa proizvodnje (eng. Computer Aided Manufacturing, skraćeno CAM). Mnogi mehanički delovi dizajniraju u paketima za 3D računarsko modelovanje, a onda automatski proizvode na glodalicama kontrolisanim od strane računara. Takođe, sa unapređenjem grafike, razvilo se i inženjerstvo potpomognuto računarstvom (eng. Computer Aided Engineering, skraćeno CAE) koje omogućava rešavanje inženjerskih zadataka uz pomoć računara.

Računarska grafika potpomaže izvođenje simulacija realnih situacija u polosvima kritičnim po sigurnost kao što su simulacije vožnje, simulacije letenja ili za trening iskusnih korisnika u situacijama koje bi bilo ili suviše skupo ili suviše opasno fizički realizovati.

Jedan od najvećih uticaja računarske grafike je svakako na industriju zabave, i to konkretno na oblast filma, televizije i igara. Film Priča o igračkama (eng. Toy Story) snimljen 1995. godine prvi je film koji je u potpunosti napravljen uz pomoć računara. Računarske igre sve više koriste sofisticirane trodimenzione modele i algoritme renderovanja, a skoro svi novi filmovi za vizuelne efekte koriste digitalni kompozit snimljene i generisane slike (eng. digital compositing), kako bi sastavili pozadinu sa zasebno snimljenim prvim planom filma. Mnogi filmovi koriste 3D modelovanje kako bi stvorili sintetička okruženja, objekte, čak i karaktere za koje većina gledalaca nikada ne bi posumnjala da nisu stvarni.

Dve podgrane računarske grafike koje su se skoro pojavile su virtuelna realnost (eng. virtual reality, VR) i proširena realnost (eng. augmented reality, AR). Virtuelna realnost je tehnologija koja omogućava simuliranje računarski generisanog okruženja sa kojim korisnik može da interaguje. Ona omogućava virtuelna putovanja u razne zemlje, virtuelne obilaske muzeja i slično, ali ima i veoma važan doprinos u svrhe treniranja, u medicini, kao i u svrhe ranije pomenutih simulacija. Za razliku od virtuelne realnosti gde je kompletan sadržaj koji se korisniku prikazuje računarski generisan, u proširenoj realnosti se korisniku prikazuje slika koja kombinuje realističan pogled na okolinu sa sadržajem koji je računarski generisan. Na ovaj način moguće je turističke ture obogatiti prikazom različitih informacija, moguće je kupcima prikazati kako bi izgledao neki artikal iz prodavnice u njihovom stanu ili čak na ovaj način isprobati šminku pre kupovine. Pre nekoliko godina veliku buru izazvala je igra za mobilni telefon

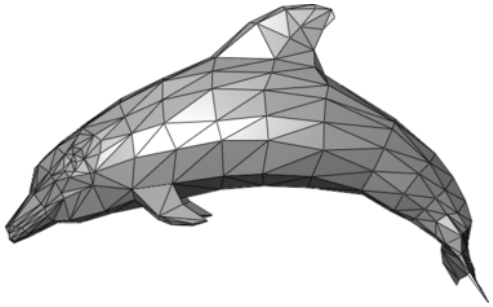
Pokemon Go koja je veliki broj mladih ljudi izvela na ulice u potrazi za virtuelnim karakterima - pokemonima koje je trebalo sakupiti.

Savremena istraživanja u oblasti računarske grafike se uglavnom bave metodama za pravljenje geometrijskih modela, metodama za realistični prikaz refleksije površi, metodama za animaciju scena prema zakonima fizike i aproksimacijama ovih zakona, interakcijom sa virtuelnim objektima i drugim. SIGGRAPH[1] je vodeća konferencija u oblasti računarske grafike; na njoj se prikazuju novi akademski rezultati iz oblasti računarske grafike, kao i komercijalni alati. Rad objavljen u zborniku radova sa ove konferencije, koji objavljuje ACM, smatra se najvažnijom referencom koju neko ko se praktično bavi ovom oblašću može da ima.

1.3 Poddiscipline računarske grafike

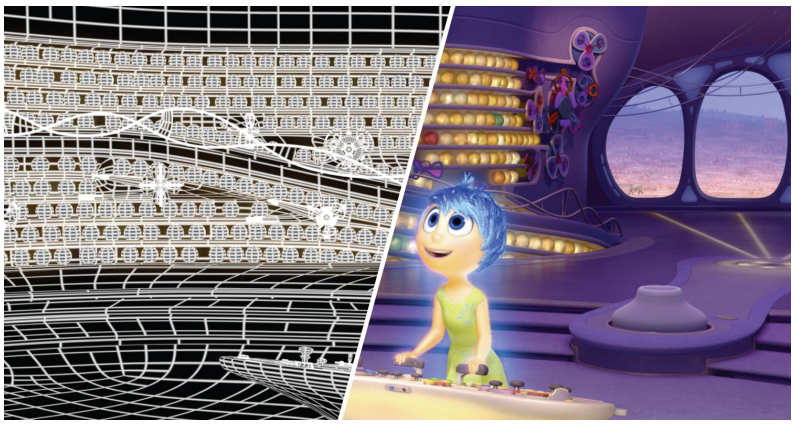
Osnovne poddiscipline računarske grafike su:

modelovanje – bavi se pre svega pravljenjem matematičke specifikacije geometrijskih objekata na sceni i njihovih vizualnih svojstava na način koji je pogodan za čuvanje na računaru (slika 1.2);



Slika 1.2: Mrežni model delfina.

renderovanje – proces kreiranja realistične dvodimenzione digitalne slike na osnovu (dvodimenzionih ili trodimenzionih) modela objekata na sceni i (realističnog ili nerealističnog) modela ponašanja svetlosti (slika 1.3);

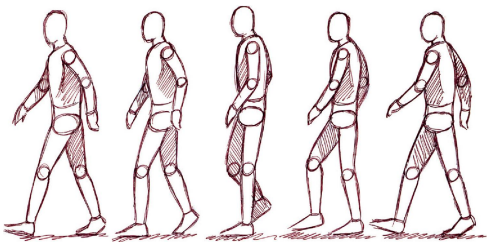


Slika 1.3: Slika iz filma Inside Out - levo je deo pre, a desno nakon renderovanja (preuzeto sa https://sciencebehindpixar.org/pipeline/rendering).

animacija – proces kreiranja nizova slika koje, kada se prikažu brzo jedna za drugom, daju utisak glatkog kretanja (slika 1.4).

Modelovanje obuhvata pravljenje modela objekata na sceni, primenu materijala na modele, postavljanje modela na scenu, modelovanje izvora svetlosti koji postoje na sceni, pozicioniranje svetla na sceni, postavljanje kamere, dok renderovanje podrazumeva pravljenje dvodimenzione slike uz pomoć kamere. U okviru ovog materijala akcenat će biti na procesima modelovanja i renderovanja.

[1] Link konferencije održane 2023. godine je https://s2023.siggraph.org/.



Slika 1.4: Niz slika ljudskog tela u pokretu kojima se stiče utisak kretanja.

Postoji veliki broj oblasti računarstva koje su u tesnoj vezi sa računarskom grafikom i koje podrazumevaju dobro poznavanje računarske grafike. Neke od njih su:

Obrada slika – bavi se zapisivanjem i obradom slika poput izdvajanja dela slike, skaliranja slike, kombinovanja više slika u jednu (slika 1.5). Pored ovog, obrada slika se bavi i rekonstrukcijom dvodimenzionih ili trodimenzionih objekata na osnovu njihovih slika. Obrada slika ima široku primenu: u satelitskim snimanjima, medicini, kosmičkim istraživanjima, robotici, prepoznavanju slova (eng. optical character recognition, OCR) itd.

Slika 1.5: Primer izdvajanja dela slike i skaliranja slike.

Virtuelna i proširena realnost – virtuelna realnost podrazumeva korišćenje računarski generisanih modela i simulacija kojima se omogućava da korisnik interaguje sa virtuelnim trodimenzionim okruženjem. Korisnik biva "ubačen" u računarski generisano okruženje koje može ili ne mora oponašati realnost i sa njim može da interaguje kroz korišćenje interaktivnih uređaja poput naočara, kaciga, rukavica i odeće kojima se šalju i primaju informacije iz okruženja simuliranog virtuelnom realnošću. Iluzija prisutnosti (teleprezentovanja) je omogućena senzorima kojima se prate korisnikovi pokreti i u skladu sa njima se u realnom vremenu slika koja se prikazuje korisniku prilagođava. Proširena realnost je korišćenje informacija u formi teksta, grafike i audio signala u realnom vremenu integrisanih sa objektima iz realnog sveta. U proširenoj realnosti se digitalni elementi dodaju realističnom pogledu korisnika, najčešće korišćenjem kamere na pametnim telefonima (slika 1.6).

3D skeniranje – koristi tehnologiju zasnovanu na pronalaženju opsega za pravljenje merljivih trodimenzionih modela (slika 1.7). 3D skeniranjem se sakupljaju informacije sa površine fizičkog objekta kojima se objekat precizno opisuje. Na ovaj način objekat je moguće digitalno analizirati, kao i iskoristiti dobijene skenirane podatke za izradu objekta na 3D štampaču. Obrada dobijenih modela često zahteva algoritme iz oblasti računarske grafike.

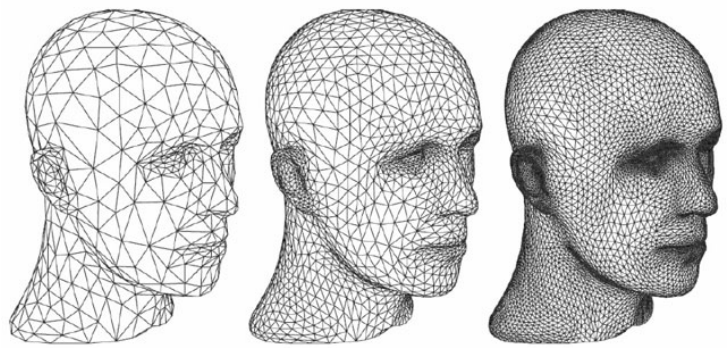
Neke od navedenih poddisciplina se međusobno prožimaju; na primer, kreiranje slika (koje spada u renderovanje) i njihova obrada imaju veliki broj zajedničkih tačaka i često je teško povući jasnu granicu između njih. Na primer, programi kao PhotoShop omogućavaju izdvajanje dela slike po nekom kriterijumu, a zatim obradu tog dela na neki zadati način.

Slika 1.6: Ilustracija virtuelne i proširene realnosti.

Slika 1.7: 3D skener za veće i manje predmete (vlasništvo MI SANU).

1.3.1 Modelovanje

Objekti koje modelujemo mogu biti jednostavniji i složeniji. Takođe, jedan isti objekat moguće je modelovati sa različitim nivoom detaljnosti[2] (slika 1.8).

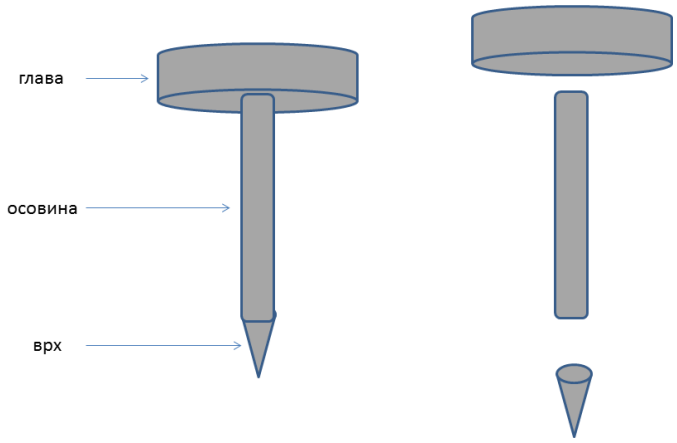


Slika 1.8: Tri različita modela istog objekta sa različitim nivoom detalja (različitim brojem trouglova u mreži).

Modeli se mogu sastojati od velikog broja geometrijskih primitiva. Neke od primitiva koje su direktno podržane u hardveru jesu tačke, duži i poligoni (trouglovi ili drugi konveksni poligoni). Pored navedenih primitiva, modelovanje takođe uključuje i korišćenje deo po deo polinomijalnih krivih, deo po deo polinomijalnih površi i slično.

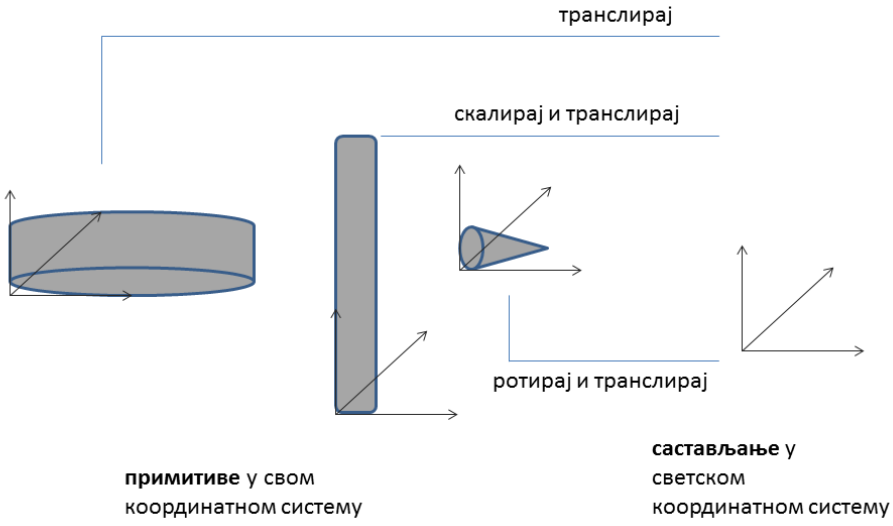
[2]U okviru projekta Digitalni Mikelanđelo koji se bavi digitalizacijom Mikelanđelovih skulptura i arhitekture (https://graphics.stanford.edu/papers/digmich_falletti/) skulptura Mikelanđelovog Davida je modelovana mrežom sačinjenom od nekoliko milijardi poligona.

Ključ modelovanja scene koja potencijalno može biti jako složena je u njenoj hijerarhijskoj strukturi. Naime, složeni objekti mogu se dobiti od jednostavnijih objekata koji se mogu napraviti od još jednostavnijih objekata i tako sve dok ne stignemo do osnovnih geometrijskih primitiva (tipa sfera, valjaka, kupa i kocki) koje se mogu direktno modelovati (slika 1.9).



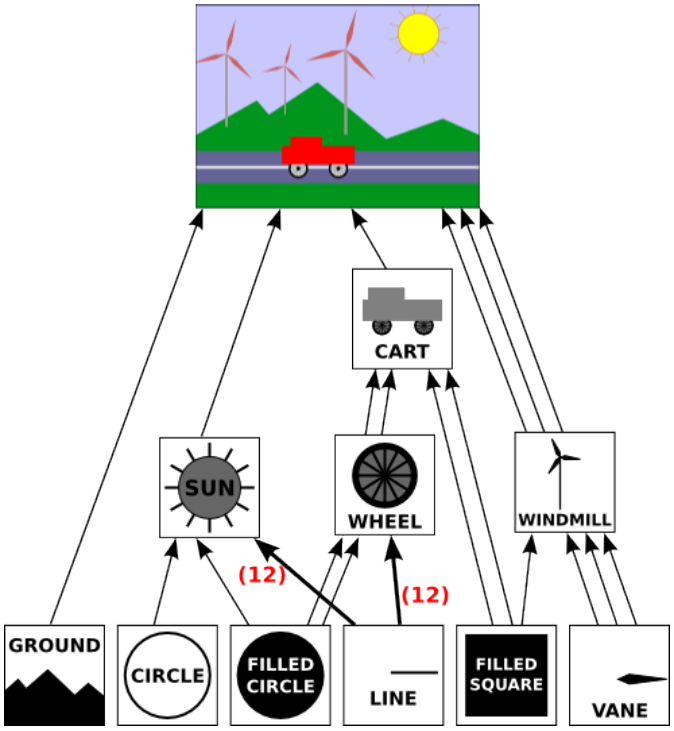
Slika 1.9: Razlaganje složenog objekta na jednostavnije komponente.

Prilikom zadavanja pojedinačnog objekta možemo koristiti koordinatni sistem u kome je objekat najjednostavnije opisati – npr. možemo pretpostaviti da je objekat centriran u koordinatnom početku i da je jedinične veličine, a zatim mu naknadno možemo postaviti veličinu, orijentaciju i poziciju na sceni primenom odgovarajućih geometrijskih transformacija, poput skaliranja, rotacija i translacija. Ove transformacije nazivamo *transformacijama modelovanja* (eng. modelling transformation). Sličan postupak možemo primeniti i na pojedinačne objekte ukoliko imaju složenu strukturu: svaki sastavni deo složenog objekta možemo da zadamo u svom pogodnom koordinatnom sistemu, a zatim ga možemo pozicionirati u okviru kompletnog objekta transformacijama modelovanja (slika 1.10). Povrh toga, možemo da primenimo drugu transformaciju modelovanja na ceo složeni objekat kao celinu da bismo ga pozicionirali unutar scene. Dakle, možemo kreirati objekte koji su sastavljeni od manjih objekata koji su ponovo sastavljeni od manjih objekata itd. Iz ovog razloga kažemo da modelovanje prati hijerarhijski princip.



Slika 1.10: Sastavljanje primitiva u polazni objekat.

Komponente složene scene formiraju hijerarhijsku strukturu podataka u kojoj je svaki objekat povezan sa svojim sastavnim delovima i ovu strukturu podataka nazivamo *graf scene* (eng. scene graph). Graf scene je drvolika struktura podataka kod koje koren predstavlja kompletnu scenu koja se renderuje, njegova deca predstavljaju objekte najvišeg nivoa na sceni, itd. Listovi grafa scene su osnovni (primitivni) objekti, kao što su sfera, kvadar i valjak, dok unutrašnji čvorovi odgovaraju primeni geometrijskih transformacija koje se zadaju matricama u homogenim koordinatama i mogu imati jedno ili više dece. S obzirom na to da čvorovi mogu biti deca većeg broja unutrašnjih čvorova, graf scene u opštem slučaju ne mora biti stablo (slika 1.11), ali jeste usmereni aciklički graf (eng. DAG, directed acyclic graph). Na ovaj način omogućeno je ponovo iskoristiti jednom definisane geometrijske oblike. Svaka od strelica na slici 1.11 se može povezati sa transformacijom modelovanja koja postavlja podobjekat na odgovarajuće mesto unutar roditeljskog objekta. U nekim aplikacijama graf scene se ne pravi eksplicitno, već se on implicitno kreira na osnovu poziva funkcija kojima se iscrtava scena.



Slika 1.11: Primer grafa scene.

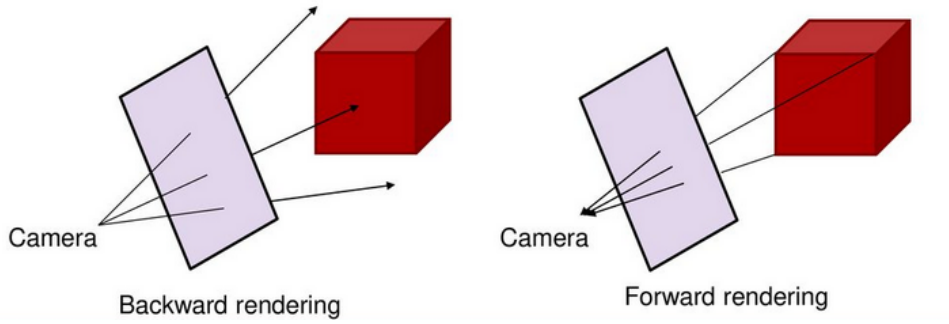
1.3.2 Renderovanje

Renderovanje predstavlja transformisanje scene sastavljene od modela objekata u trodimenzionom prostoru u dvodimenzionu sliku. Modeli su sastavljeni od primitiva koje su podržane u sistemu za renderovanje.

Razlikujemo dve vrste renderovanja: renderovanje unapred i renderovanje unazad. Kod renderovanja unapred, elementi scene se projektuju ka kameri. Naime, za svaku primitivu potrebno je utvrditi u koje se piksele projektuje i ovaj postupak naziva se *rasterizacija*. Tehnika rasterizacije je veoma efikasna – njom je korišćenjem savremenog hardvera moguće prikazati milijarde primitiva (prevashodno trouglova) u sekundi, ali je teže na slici postići fotorealizam, senke i refleksije koje možemo videti okom. Ovaj proces renderovanja je najčešće podržan u hardveru.

S druge strane, tehnika praćenja zraka (eng. ray tracing), koju ćemo izložiti naknadno u okviru kursa, predstavlja primer algoritma renderovanja unazad kod koga se kreće od tačke na slici i onda se utvrđuje koje se primitive projektuju na nju (slika 1.12). Ova tehnika je u opštem slučaju sporija od rasterizacije (ponekad je potrebno sačekati minute ili sate da bismo generisali jednu sliku), ali je njom lakše postići fotorealizam.

Tradicionalno, transformacija iz modela preko scene pa sve do slike se "razbija" na manje korake, koji se nazivaju *grafičkom protočnom obradom* (eng. graphics pipeline). Dakle, grafička protočna obrada je osnovni okvir u računarskoj grafici kojim se zadaju koraci prilikom transformacije trodimenzione scene u dvodimenzionu reprezentaciju na ekranu. Svaki od koraka u grafičkoj protočnoj obradi prosleđuje



Slika 1.12: Renderovanje unazad i renderovanje unapred (preuzeto sa https://slideplayer.com/slide/13550988/).

rezultat svoje faze narednom koraku. Ona se može implementirati na različite načine i neki od koraka se mogu implementirati u samom hardveru, a neki softverski. S obzirom da je čitav proces podeljen na striktno definisane korake, svaki od koraka grafičke protočne obrade može se zasebno optimizovati. U ranim fazama razvoja računarske grafike, grafička protočna obrada bila je *fiksna*, odnosno programer nije mogao da utiče na izlaz nijedne od faza protočne obrade. Danas je situacija drugačija i neke od faza je moguće kontrolisati korišćenjem programa za grafički procesor koji se nazivaju *šejderi* (eng. shader). Ova vrsta grafičke protočne obrade se naziva *programabilna protočna obrada*. Revolucionarni napredak u oblasti računarske grafike desio se pojavom šejdera koji su omogućili finiju kontrolu grafičke protočne obrade.

1.4 Podele računarske grafike

Prema vremenu u kojem se vrši, postoje dve vrste renderovanja: *renderovanje u realnom vremenu* (koje se nekada naziva i *onlajn renderovanje*) i *oflajn renderovanje*. Renderovanje u realnom vremenu je tehnika koja omogućava kreiranje 3D slika i animacija u realnom vremenu, kao što je slučaj u računarskim igrama. Oflajn renderovanje se koristi za kreiranje realističnih slika i filmova, kada je dozvoljeno da se svaka slika renderuje i po nekoliko sati ili dana. Ono se najčešće koristi u filmskoj industriji za renderovanje realističnih scena visokog kvaliteta i za specijalne efekte.

Takođe, grafiku možemo podeliti na interaktivnu računarsku grafiku i onu koja to nije. *Interaktivna računarska grafika* podrazumeva dinamički način prikaza slike na računaru uz aktivno učešće čoveka u stvaranju i izmeni slike, gde su rezultati odmah vidljivi, kao što je slučaj kod računarskih igara ili simulacija letenja. Nasuprot tome, pod *neinteraktivnom računarskom grafikom* podrazumeva se svako generisanje ili predstavljanje slikovnih informacija koje ne zadovoljava prethodne uslove; u ovom slučaju se, dakle, radi o statičnim informacijama, koje su predstavljene bojom i oblikom, bez interakcije čoveka (npr. animacije i filmovi).

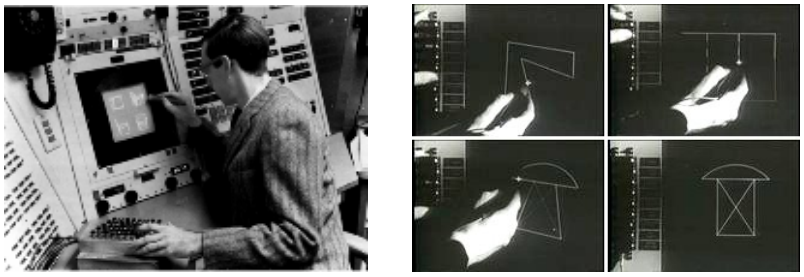
1.5 Interaktivna računarska grafika – od Sketchpad-a do danas

U interaktivnoj računarskoj grafici korisnik kontroliše sadržaj, strukturu i izgled objekata i njihovih slika koje se prikazuju korišćenjem brze vizualne povratne informacije. To se postiže korišćenjem različitih ulaznih uređaja.

1.5.1 Počeci interaktivne grafike

> "Sistem Sketchpad koristi crtanje kao novi vid komunikacije sa računarom. Sistem sadrži ulaz, izlaz i računarske programe koji omogućavaju interpretaciju informacija koje se direktno iscrtavaju na računarskom ekranu. Sketchpad se pokazao korisnim kao pomoć u razumevanju procesa, kao što je kretanje veza, koje se može opisati slikama. Sketchpad, takođe, olakšava iscrtavanje slika koje sadrže veliki broj ponavljenih elemenata i slika koje treba da budu jako precizne, kao i izmenu slika prethodno iscrtanih pomoću njega."
>
> Ivan Sutherland, prvi pasus doktorske teze "Sketchpad, A Man-Machine Graphical Communication System", 1963, MIT.

Sketchpad[3] je prvi u potpunosti interaktivni grafički sistem, razvijen u MIT-ovim laboratorijama početkom sedamdesetih godina prošlog veka. On je omogućavao komunikaciju sa računarom crtanjem po ekranu računara (slika 1.13). Sketchpad je uveo mnoge koncepte koji postoje i u današnjim sistemima, kao što su kontekstni meniji, hijerarhijsko modelovanje i slično. Omogućavao je jednostavno iscrtavanje slika koje imaju veliki broj ponovljenih elemenata, kao i jednostavnu izmenu slika prethodno iscrtanih pomoću njega. Doktorska teza Ivana Saderlanda (Ivan Sutherland) predstavlja prvu doktorsku tezu iz oblasti računarske grafike.



Slika 1.13: Sketchpad.

Interaktivna računarska grafika je dugo ostala van mogućnosti svih osim tehnološki najrazvijenijih organizacija. Neki od razloga za to su predstavljale visoke cene grafičkog hardvera (koji se proizvodio u malim serijama), neportabilan softver koji je najčešće pisan za poseban proizvođačev uređaj za prikaz, nepostojanje grafičkih standarda i sl. Danas je interaktivna računarska grafika prisutna u svakodnevnom životu svakog pojedinca: od ekrana računara, preko ekrana pametnih telefona, pa sve do računarskih igara.

1.5.2 Razvoj interfejsa za rad sa računarima

Najraniji računari su bili veoma glomazni i imali su jako komplikovan interfejs za rad. Naime, jedan od prvih vidova komunikacije sa računarom predstavljale su bušene kartice – računari su putem bušenih kartica primali informacije i instrukcije od korisnika, a, takođe, na isti način korisniku prikazivali rezultat svog rada. Ovakav vid komunikacije je iziskivao puno vremena i umešnosti korisnika kako za pripremu podataka, tako i za interpretiranje dobijenih rezultata. Bilo je jasno da je neophodno da se ta komunikacija unapredi. Tokom narednih godina računari su dobijali neke vidove ekrana, putem kojih je bilo moguće prikazati informaciju koja je dobijena kao rezultat, a s godinama razvoj uređaja za prikaz, kao i ulaznih uređaja je bivao sve savremeniji i ugodniji.

Razvoj ulaznih uređaja

Za upravljanje računarom koriste se ulazni uređaji koji prihvataju podatke i instrukcije od korisnika i konvertuju ih u formu razumljivu računaru. Do nedavno se komunikacija od korisnika ka računaru uobičajeno realizovala uređajima poput miša i tastature. Međutim, poslednjih godina došlo je do značajnog unapređenja ulaznih uređaja. Ulazni uređaji se grubo mogu podeliti na:

- *tekstualne ulazne uređaje:* npr. tastatura,
- *pokazivačke uređaje:* miš, trackball, ekran osetljiv na dodir (slika 1.14), olovka, tj. stajlus (eng. stylus),
- *haptičke uređaje*: džojstik, volan za upravljanje i drugi kontroleri igara (kao što je Wii) - njima se simuliraju određeni fizički atributi, čime se omogućava da korisnik dodirom direktno komunicira sa virtuelnim objektima; poseban tip haptičkih uređaja predstavljaju uređaji zasnovani na povratnoj sprezi (eng. force-feedback) kojima se korisniku vraća i određena povratna informacija, kao što je otpor prilikom kretanja, vibracija ili efekat opruge kojim se džojstik vraća u početnu poziciju (slika 1.15).
- *slikovne i video uređaje:* skener, digitalni fotoaparat, digitalna kamera, . . .

[3] Više o sistemu Sketchpad možete pogledati na https://www.youtube.com/watch?v=fwzYuhduME4.

Slika 1.14: Multi-touch – funkcionalnost pomoću koje ekrani osetljivi na dodir registruju istovremeni višestruki dodir po površini (preuzeto sa: https://www.engadget.com/2010/04/20/synaptics-extends-multitouch-gesture-suite-to-linux-chrome-os-i/).

Slika 1.15: Radna stanica zasnovana na povratnoj sprezi koja predstavlja integrisani sistem simulacija pokreta kompletne šake leve i desne ruke (preuzeto sa: http://www.diytrade.com/china/pd/5554912/Immersion_Haptic_Force_Feedback_Workstation.html).

- *senzorske uređaje:* kinekt (eng. kinect) je ulazni uređaj za praćenje pokreta proizveden od strane kompanije Microsoft, originalno za potrebe eliminisanja kontrolera igara za XBox video igre; američka kompanija Leap Motion bavi se proizvodnjom i prodajom senzorskih uređaja koji pružaju podršku pokretima prstiju i šake, slično mišu i predstavljaju ulaznu tehnologiju koja radi bez fizičkog kontakta[4].

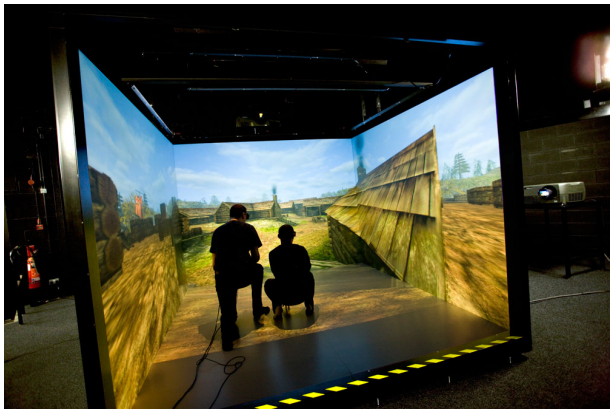
Poslednjih godina primećuje se nagli trend razvoja tehnologije sa namerom unapređenja doživljaja virtuelne realnosti. Jedan od takvih uređaja jesu i HMD (eng. Head-Mounted Displays) uređaji za prikaz koji imaju mali optički uređaj ispred jednog ili oba oka (slika 1.16). Najčešće se njima prikazuju samo računarski generisane slike, a poneke varijante mogu da kombinuju i pogled na realni svet.

Slika 1.16: HMD uređaji (preuzeto sa: https://en.wikipedia.org/wiki/Head-mounted_display).

Razvijaju se i trodimenzioni prostori virtuelne stvarnosti, poput virtuelnih pećina (eng. cave) [5]. Virtuelna pećina je okruženje virtuelne stvarnosti u kojoj su projektori usmereni na zidove kocke veličine sobe. Zidovi su uobičajeno napravljeni od projektorskih ekrana, a pod, takođe, može da bude projektorski ekran ili tanki panel za prikaz. Sistemi za projektovanje su veoma visoke rezolucije kako bi obezbedili uverljivu iluziju stvarnosti. Korisnici nose 3D naočare i mogu da vide objekte koji lebde u

[4]Prikaz senzorskog uređaja je dostupan na adresi: https://www.youtube.com/watch?v=zXghYjh6Gro&feature=youtu.be.
[5]Simulacija pećine dosupna je na adresi https://www.youtube.com/watch?v=j59JxfbvxGg.

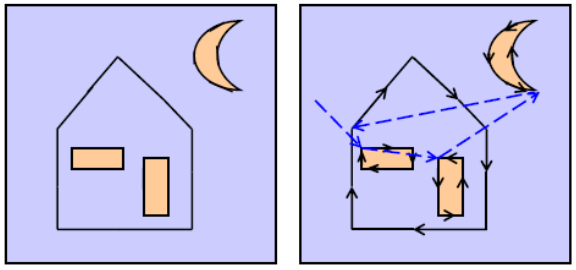
vazduhu, mogu da se kreću oko njih i da sa njima interaguju (slika 1.17). Prostor je opremljen i velikim brojem zvučnika postavljenih pod različitim uglovima kojima se trodimenzioni video doživljaj dopunjuje zvučnim efektima.



Slika 1.17: 3D virtuelne pećine (preuzeto sa: http://147.91.14.147/Blog.aspx?id=252).

Razvoj izlaznih uređaja

Na samom početku razvoja uređaja za prikaz, od 1950-ih do sredine 1970-ih, korišćeni su *vektorski sistemi*. Kod njih se linija dobija tako što se digitalne koordinate krajnjih tačaka transformišu u analogni napon za elektronski zrak koji pada na površinu ekrana (slika 1.18). Ova metodologija se zove i *random scan* metodologija (zato što linija može da spaja „bilo koje dve tačke na ekranu“). Slika se na vektorskim sistemima osvežava obično 30 do 60 puta u sekundi (30-60 Hz) i moguće je iscrtati i do 100000 linija. Pritom su linije glatke, ali se obojene površine teško prikazuju. Vektorski sistemi su pogodni za mrežne modele. Vektorski ekrani su imali veoma visoku rezoluciju (neki ekrani su mogli da adresiraju i do 4000×4000 tačaka). Vektorski monitori ne prate nužno sadržaj čitavog ekrana, već samo duži od kojih se sastoji slika koja se prikazuje, te brzina osvežavanja slike zavisi od kompleksnosti podataka od kojih se slika sastoji.



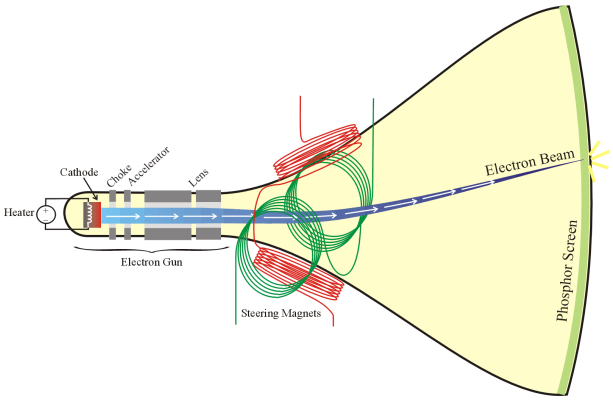
Slika 1.18: Iscrtavanje slike na vektorskom monitoru.

1960-ih se javlja nova generacija vektorskih sistema, pod nazivom *DVST sistemi* (eng. direct view storage tube). Kod njih se ne zahteva velika učestalost osvežavanja slike (jer je slika predstavljena raspodelom naelektrisanja na unutrašnjoj površini ekrana). Međutim, prilikom izmene i najmanjeg detalja na slici, kompletna slika (raspodela naelektrisanja) mora ponovo biti kreirana (slika 1.19).

1970-ih se javljaju *CRT sistemi* (eng. cathode ray tube) koji predstavljaju *rastersku grafiku* zasnovanu na televizijskoj tehnologiji i katodnim cevima. Kod ovih sistema slika je predstavljena pikselima. Oni funkcionišu po sledećem principu: elektronski top emituje zrak elektrona koji se ubrzava pozitivnim naponom do fosforom presvučenog ekrana. Na putu do ekrana, mehanizmom za fokusiranje elektroni se usmeravaju ka konkretnoj tački na ekranu (slika 1.20). S obzirom na to da fosfor emituje svetlost koja opada eksponencijalno sa vremenom, sliku je potrebno osvežavati, najčešće 60 puta u sekundi (60Hz).

U ovim sistemima piksel nema jasno određene ivice već se za njegovu veličinu uzima prečnik oblasti gde je intenzitet emitovanja veći od 50% intenziteta u središtu oblasti. Kod CRT sistema sadržaj slike ne utiče na brzinu prikazivanja, te u nekim situacijama kreiranje slike može da bude zahtevnije nego na vektorskim sistemima: na primer, u vektorskom sistemu za telo koje rotira dovoljno je preračunavati koordinate kontrolnih tačaka.

Slika 1.19: Tektronix računar sa DVST monitorom (1970-e).



Slika 1.20: Ilustracija rada katodne cevi.

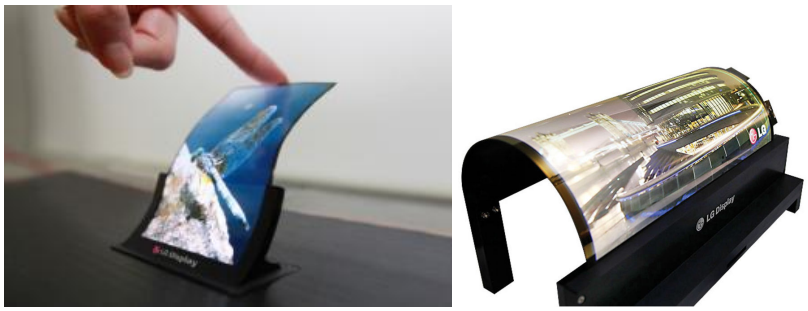
Za razliku od vektorskih sistema kod kojih su linije glatke, u rasterskim sistemima kose linije su „stepenaste“ (slika 1.21), ali se zato obojene površine lakše prikazuju (problem popunjavanja regiona nije težak).

Slika 1.21: Iscrtavanje slike i bojenje površina na rasterskom monitoru.

CRT monitori se i danas neretko koriste u štamparskoj industriji, medijskim kompanijama, oblasti profesionalne fotografije – jer daju pouzdanije boje i kontrast, kao i širi ugao posmatranja.

LCD ekrani (eng. liquid-crystal display) postaju široko rasprostranjeni od sredine 1990-ih. Oni predstavljaju tanke, ravne prikazne uređaje. Zasnovani su na korišćenju dugačkih molekula kristala, koji zahvaljujući specifičnom korišćenju elektriciteta i polarizacije svetlosti stvaraju osvetljene ili tamne delove ekrana. LCD paneli ne proizvode svetlost, te im je neophodan izvor svetla (tzv. ”pozadinsko svetlo“). Glavna prednost im je što troše veoma malo električne energije. Slika se kod LCD ekrana osvežava i do 200 puta u sekundi (200Hz). Za razliku od CRT ekrana, LCD ekrani imaju nativnu fiksnu rezoluciju, sa fiksnim rasterom.

Kao podtip LCD ekrana postoje i LCD ekrani sa aktivnom matricom (eng. active matrix) kod kojih je u svaki piksel ugrađen po jedan tranzistor (koji opisuje taj piksel), te kod njih nema treperenja i boje su sjajnije. Kod ekrana sa aktivnom matricom se slika ažurira brže nego kod onih sa pasivnom matricom i ovakvi ekrani imaju širi vidni ugao. Jedan od podtipova LCD ekrana sa aktivnom matricom su *TFT ekrani* (eng. Thin Film Transistor). Kod TFT ekrana se nekoliko puta u sekundi vrši kontrola individualnih piksela kako bi se utvrdilo da li ima promena.



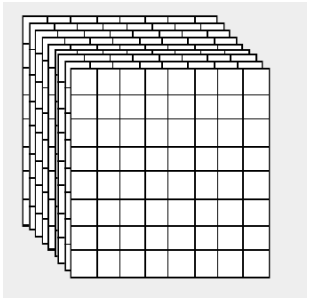
Slika 1.22: Fleksibilni OLED ekrani.

U novije vreme javljaju se i *OLED ekrani* (eng. organic light-emitting diode). Oni se prave od organskih materijala koji emituju svetlost kada kroz njih prolazi elektricitet. Dok je kod LCD ekrana potrebno da postoji pozadinsko svetlo, pikseli kod OLED ekrana proizvode svoju svetlost, te su oni efikasniji i dosta tanji i podloga OLED ekrana ne mora biti kruta već može biti i fleksibilna (slika 1.22). Takođe na OLED ekranima moguće je postići veću kontrast nego kod LCD ekrana. Međutim ova tehnologija je dosta složenija i zbog toga su OLED ekrani značajno skuplji od LCD ekrana.

1.6 Arhitektura rasterskih sistema za prikaz

Osnovne komponente rasterskog sistema za prikaz čine procesor za prikaz, odnosno grafički procesor (eng. display processor, graphics processing unit, GPU), video kontroler (eng. video controler), frejm bafer (eng. frame buffer) i ekran.

U rasterskim sistemima slika se formira u *frejm baferu* procesom rasterizacije. Na frejm bafer se može gledati kao na računarsku memoriju organizovanu u vidu dvodimenzionog niza tako da svaka adresibilna lokacija (x, y) odgovara jednom pikselu u kome se čuva njegova boja. *Bitska dubina* (eng. bit depth) je broj bitova koji odgovara svakom pikselu u frejm baferu (slika 1.23). U frejm baferu se čuva sadržaj koji se zatim prikazuje na ekranu. Prilikom osvežavanja sadržaja ekrana, posmatraču svaki naredni frejm izgleda kao nastavak prethodno prikazanog kadra i slike se smenjuju glatko ukoliko učestalost osvežavanja slike nije previše niska. Standardna učestalost osvežavanja slike ekrana računara je 60 Hz, ali se poslednjih godina razvijaju i specijalizovani monitori koji imaju i osvežavanje od 120 Hz i 144 Hz.



Slika 1.23: Vizualizacija frejm bafera.

Grafički procesor je hardver specijalne namene razvijen da omogući efikasnu rasterizaciju grafičkih primitiva koje se smeštaju u frejm bafer. Moderni grafički procesori su zahvaljujući svojoj arhitekturi daleko efikasniji od procesora opšte namene u izvršavanju matematičkih operacija fundamentalnih za iscrtavanje slike.

Video kontroler je uređaj koji upravlja taktom osvežavanja piksela na ekranu. U svakom osvežavanju video kontroler čita sadržaj video memorije i postavlja boje piksela na ekranu tako da najbliže odgovaraju slici čiji je sadržaj predstavljen vrednostima video memorije.

Aplikacija izdaje naredbe za iscrtavanje grafičkom procesoru pomoću API-ja definisanog standardom. Najkorišćeniji standardi za iscrtavanje su OpenGL, DirectX, WebGL, Vulkan i Metal. Standardom se definiše skup funkcija i njihovo ponašanje pomoću kojih programer konstruiše deo programa koji je zadužen za iscrtavanje. Standardom se opisuje kako funkcije treba da rade, ali nije garancija da će raditi isto na svakoj grafičkoj kartici koja dati standard implementira, što nam iskustvo svedoči. Grafičke bib-

lioteke imaju podršku za zadavanje primitiva poput karaktera, pravih, poligona, za zadavanje atributa poput boje, stila linije, svojstva materijala za 3D i slično, za zadavanje svetlosti, geometrijskih transformacija i dr.

Grafička biblioteka se nalazi između aplikacije i hardvera za prikaz (grafičkog sistema). Program aplikacije preslikava objekte aplikacije u njihove slike pozivom funkcija grafičke biblioteke. Prilikom interakcije sa korisnikom vrši se izmena slike i/ili modela.

Po tome da li se scena iscrtava iznova ili delimično razlikujemo *direktni* (eng. immediate regime) i *odloženi režim* (eng. retained regime) renderovanja. Igrice i simulacije koriste direktan režim jer se većina piksela razlikuje iz frejma u frejm, a grafički korisnički interfejsi najčešće koriste odloženi režim jer većina piksela ostaje nepromenjena iz frejma u frejm. Za odloženi režim neophodno je predstaviti scenu u vidu grafa scene kako bi događaji i promene na sceni bile propagirane samo onim delovima scene na kojima je došlo do promene.

1.7 Pitanja

1.1 Čime se bavi računarska grafika?

1.2 Navesti bar četiri različita domena u kojima se koristi računarska grafika.

1.3 Koje su osnovne poddiscipline računarske grafike?

1.4 Koja je razlika između virtualne i proširene realnosti?

1.5 Koji su osnovni zadaci procesa modelovanja?

1.6 Šta se podrazumeva pod tim da je proces modelovanja hijerarhijski?

1.7 Čime se bavi renderovanje?

1.8 Koja je razlika između renderovanja unapred i renderovanja unazad?

1.9 Šta razlikuje interaktivnu od neinteraktivne računarske grafike?

1.10 Kako se zove prvi interaktivni grafički sistem? Ko je njegov autor?

1.11 Na koji način se generiše slika kod vektorskih sistema?

1.12 Kakav izgled imaju kose linije u vektorskim, a kakav u rasterskim sistemima?

1.13 Čemu služi frejm bafer? Čemu služi video kontroler?

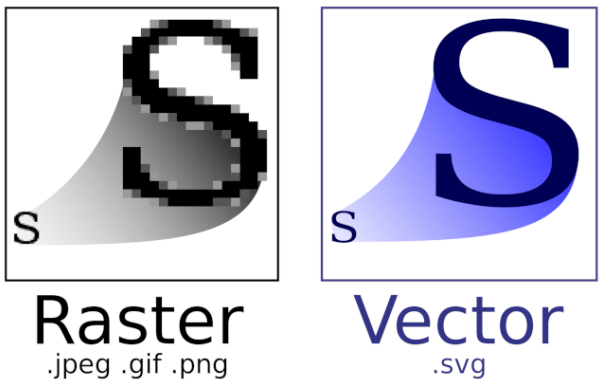
1.14 Koji je osnovni zadatak grafičkog procesora?

1.15 Koja je razlika između direktnog i odloženog režima renderovanja?

Glava 2

Rasterizacija

Postoje dve osnovne vrste zapisa grafičkog sadržaja: rasterski i vektorski (slika 2.1). Rasterska grafika koristi diskretne uzorke za opisivanje vizuelne informacije, dok se u vektorskoj grafici čuvaju matematički opisi ili modeli geometrijskih elemenata (pravih, krivih, . . .) i pridruženih atributa.



Slika 2.1: Rasterska i vektorska slika.

Rasterska slika je predstavljena kao pravougaona mreža elemenata koje nazivamo *pikseli* (eng. pixel, picture element). Oni za neku lokaciju tačke na slici čuvaju informaciju o boji, intenzitetu, osvetljenosti, transparentnosti i dr. Rasterske slike se dobijaju digitalnim fotoaparatima, skeniranjem slika ili odgovarajućim softverom za kreiranje rasterskih slika (na primer Adobe Photoshop). Svaka slika sadrži fiksiran broj piksela i broj piksela definiše kvalitet slike i on odgovara rezoluciji slike. Ako bi slika sadržala veći broj piksela, to bi dovelo do boljeg kvaliteta slike iste ili veće dimenzije, ali bi, takođe, povećalo veličinu odgovarajuće datoteke. Najpoznatiji rasterski formati su bmp, gif, jpg i png. U tabeli 2.1 dat je uporedan pogled najčešće korišćenih rasterskih formata.

Informacija o boji svakog piksela čuva se u nizu bitova fiksne dužine. Kao što smo već pomenuli broj bitova koji se koristi po pikselu naziva se bitska dubina (dubina piksela, odnosno dubina boje). Što je bitska dubina veća, moguće je prikazati veći broj boja na slici. Postoji veliki broj standarda za čuvanje boje piksela, od kojih su neki bitmapa, nijanse sive, RGB, RGBA, CMYK, . . .

Crno-bela slika ili bitmapa koristi 1 bit po pikselu i kod nje je svakom pikselu dodeljena jedna od dve vrednosti: nula (crna) ili jedan (bela). U slikama koje podržavaju nijanse sive se svakom pikselu pridružuje 8 bitova, odnosno jedan od 256 različitih nivoa intenziteta. RGB je veoma široko rasprostranjen režim boja digitalnih slika koji koristi tri kanala za predstavljanje crvene, zelene i plave boje.

	JPG	PNG	GIF
kompresija bez gubitka	NE	DA	DA
podržana transparentnost	NE	DA	DA
broj podržanih boja	24-bitne	24-bitne	8-bitne
podržana animacija	NE	NE	DA

Tabela 2.1: Karakteristike osnovnih rasterskih formata.

Koriste ga slike u različitim formatima: .jpg, .png, .gif, .bmp. RGBA je standard koji pored informacije o crvenoj, zelenoj i plavoj komponenti boje čuva i informaciju o transparentnosti kroz dodatni alfa kanal. Ovaj standard je karakterističan za .png i .gif formate slika koji imaju podršku za transparentnost. CMYK je standard boja koji se koristi za štampanje, gde se svakom od piksela slike dodeljuje procenat svakog od četiri mastila i koji koristi 32 bita po pikselu.

Slika se u vektorskom formatu čuva kao kolekcija geometrijskih primitiva kao što su tačke, linije, krive i poligoni, a njihova reprezentacija se temelji na matematičkim jednačinama. Za svaku od primitiva zadaje se gde će se nalaziti na slici.

Razmotrimo kako bi izgledao zapis vektorske, a kako rasterske slike koja sadrži jedan krug. Za čuvanje vektorske slike kruga potrebno je zapamtiti njegov poluprečnik, koordinate centra kruga, stil i boju linije i boju unutrašnjosti kruga, dok bi u slučaju rasterske slike kruga bilo potrebno zapamtiti vrednosti boje za sve piksele slike. Dakle, u slučaju kruga bi za vektorsku sliku bilo potrebno sačuvati dosta manje informacija nego za odgovarajuću rastersku sliku, te bi veličina odgovarajuće datoteke bila manja. Međutim, memorija koju vektorska slika zauzima zavisi od njene složenosti: memorija koju zauzima vektorska slika koja sadrži samo jedan krug razlikovala bi se od toga koliko memorije bi zauzimala vektorska slika iste dimenzije koja sadrži na hiljade krugova.

Jedna od prednosti rasterske grafike u odnosu na vektorsku grafiku je da kada se slika jednom definiše u terminima boje na (x, y) poziciji mreže, ona se lako može modifikovati izmenom lokacije ili vrednosti boje, informacije o pikselima jedne slike se mogu iskopirati u drugu, zamenom ili kombinovanjem sa prethodnim pikselima. Mana ovog pristupa je ta što ne postoje dodatne informacije o slici već za grafiku zasnovanu na uzorku važi paradigma WYSIAYG (skraćenica od eng. What You See Is All You Get). Dakle, ne postoje informacije o dubini objekta na slici, niti se scena može istraživati iz druge tačke pogleda.

Rastersku sliku moguće je skalirati, međutim, smanjenjem slike u odnosu na izvornu dimenziju gube se informacije o slici. Na primer, ako sliku smanjimo na 50% originalne dimenzije biće prikazan svaki drugi piksel, te slika postaje nazubljena. Povećavanjem slike u odnosu na izvornu dimenziju gube se fini detalji i slika postaje mutna (slika 2.2 levo). Rezolucijom slike je ograničeno do koje dimenzije je moguće povećati sliku a da ne vidimo pojedinačne piksele. Međutim, ako bismo imali sliku male dimenzije u visokoj rezoluciji, to bi dovelo do spajanja velikog broja piksela i vizuelnog efekta nalik onom kada slika velike dimenzije nije dovoljno velikog kvaliteta.

Slika 2.2: Skaliranje rasterske i vektorske slike (preuzeto sa https://guides.lib.umich.edu/c.php?g=282942&p=1885352).

Vektorsku sliku je moguće skalirati (povećati, smanjiti) bez gubitka na kvalitetu slike (slika 2.2 desno). Takođe, pomeranje elemenata na slici i njihovo popunjavanje ne bi negativno uticalo na kvalitet slike kao u slučaju rasterske grafike. Ipak, vektorska grafika nije pogodna za predstavljanje fotografija i fotorealističnih slika zbog nesavršenosti koje su neminovno prisutne u realnom svetu i koje onemogućuju opisivanje realnog sveta u vektorskom formatu. Vektorska grafika se prevashodno koristi tamo gde je potrebno omogućiti skaliranje bez gubitka na kvalitetu slike, poput fontova, logotipova kompanija i slika koje u sebi sadrže tekst.

S obzirom na to da je prikaz slike na današnjim ekranima u vidu dvodimenzione mreže piksela, da bi se slika prikazala, bez obzira na njen format, mora biti prevedena u rasterski oblik.

2.1 Rasterizacija trougla

Proces transformisanja osnovnih objekata niskog nivoa u odgovarajući rasterski oblik, tj. u reprezentaciju pikselima naziva se *rasterizacija* (eng. rasterization). S obzirom na to da trouglovi predstavljaju osnovnu grafičku primitivu, mi ćemo se u ovom poglavlju baviti problemom rasterizacije trougla.

3D primitive koje se nalaze na sceni su najčešće predstavljene skupom temena i skupom trouglova i one predstavljaju ulaz u proces rasterizacije, dok je kao izlaz potrebno dobiti rastersku sliku koja za svaki piksel čuva vrednost boje, potencijalno njegove dubine i transparentnosti. Rasterizacija se po

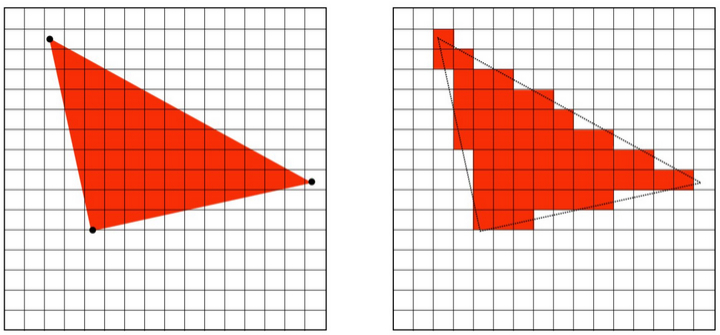
pravilu izvodi korišćenjem grafičkog procesora koji je specijalizovan za tu namenu i zbog toga se ta operacija izvodi veoma efikasno.

Postavlja se pitanje zašto su baš trouglovi osnovna grafička primitiva. Najpre, trougao je najjednostavnija geometrijska figura koja ima površinu. Pokazuje se da je proizvoljno složenu figuru moguće predstaviti trouglovima, ukoliko na raspolaganju imamo dovoljno veliki broj trouglova. Naravno, neke primitive je moguće verno predstaviti na ovaj način, a neke dovoljno dobro aproksimirati. Ono što je veoma značajna karakteristika trougla jeste da on uvek leži u ravni, što je veoma važno u situacijama kada nam je, kao npr. kod problema senčenja, potrebno poznavanje vektora normale u nekoj tački – u ovom slučaju vektor normale biće dobro definisan i konstantan za sve tačke trougla. Ukoliko su vrednosti nekog atributa zadate u temenima, one se lako mogu interpolirati za sve tačke u unutrašnjosti trougla korišćenjem baricentričnih koordinata. Konačno, kada sve svedemo na trouglove, moguće je optimizovati operacije za rad nad trouglovima.

Grafička protočna obrada za proces rasterizacije se sastoji iz narednih koraka:

- projekcije objekata na dvodimenzioni ekran
- za svaki trougao potrebno je utvrditi koje piksele prekriva
- interpolacije atributa datih u temenima trougla na tačke u unutrašnjosti trougla
- izračunavanje boje svakog piksela
- za svaki piksel kombinovanje vrednosti za svaki od trouglova kako bi se dobila konačna slika.

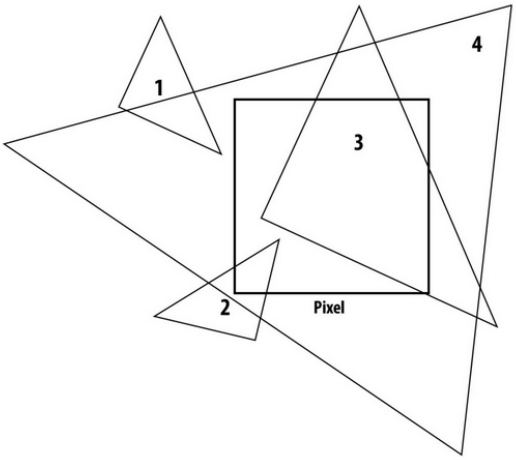
Kao što smo već pomenuli jedan od važnih zadataka grafičke protočne obrade jeste i za dati trougao utvrditi koje piksele prekriva. Ulaz u ovaj korak čine projektovane dvodimenzione koordinate temena trougla, dok bi na izlazu želeli da za svaki piksel odredimo binarnu vrednost koja nam govori da li je taj piksel prekriven datim trouglom ili ne, pri čemu vrednost 0 označava da piksel nije prekriven, a 1 da jeste (slika 2.3). Naravno, postavlja se pitanje za koje piksele treba zaključiti da su prekriveni datim trouglom. Naime, piksel ima svoju površinu i može se desiti da se deo trougla nalazi unutar piksela, a deo van. Razmotrimo ilustraciju prikazanu na slici 2.4: za trougao 1 bismo jednostavno zaključili da ne prekriva dati piksel jer je sa njim disjunktan, za trougao 4 da ga prekriva jer se kompletan piksel nalazi u njegovoj unutrašnjosti, međutim, za trouglova 2 i 3 ne bismo mogli da sa izvesnošću tvrdimo da li prekrivaju dati piksel. Stoga se pokazuje da je umesto binarne vrednosti pogodnije izračunati procenat površine piksela koji je prekriven trouglom i obojiti piksel nijansom koja odgovara tom procentu. Računanje prekrivenosti postaje mnogo složenije kada razmatramo realističnu scenu koja sadrži veliki broj trouglova i kada se može desiti da dva različita trougla prekrivaju jedan isti piksel, kada imamo situaciju da jedan trougao zaklanja drugi, kad u igru uđu transparentni objekti, ili kada dobijemo nekonveksne regione za koje je jako teško izračunati tačan procenat prekrivenosti.



Slika 2.3: Vrednosti piksela koje odgovaraju funkciji prekrivenosti piskela trouglom – crveno obojeni piksel označava da je piksel prekriven, a beli da nije (preuzeto iz kursa Univerziteta Carnegie Mellone).

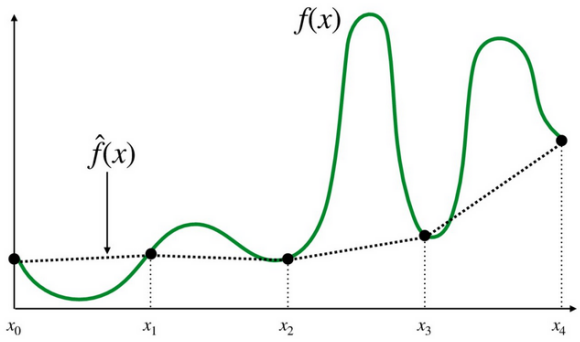
Naime, pokazuje se da računanje tačne vrednosti prekrivenosti piksela trouglom nije praktično. Iz tog razloga ćemo ovaj problem razmatrati kao problem *uzorkovanja* (eng. sampling), tako što ćemo za svaki uzorak sa slike utvrditi koji je trougao u toj tački vidljiv. Pokazuje se da sa dovoljno velikim brojem tačaka i pametnim izborom lokacija tačaka možemo na ovaj način dobro aproksimirati procenat prekrivenosti piskela trouglom.

U opštem slučaju uzorkovanje funkcije podrazumeva merenje vrednosti funkcije u nekim tačkama. Na taj način mi transformišemo neprekidni signal u digitalni oblik. Pogodan primer na kome možemo



Slika 2.4: Različiti položaji trouglova u odnosu na piksel (preuzeto iz kursa Univerziteta Carnegie Mellone).

ilustrovati ovu tehniku jeste audio fajl. Na primer, audio fajl čuva uzorkovane vrednosti jednodimenzionog signala pri čemu se uzorkovanje vrši 44.100 puta u sekundi. Audio signal u stvari čuva vrednosti amplitude tokom vremena. Nakon konvertovanja audio signala u niz vrednosti, potrebno je rešiti i inverzni problem – na koji način na osnovu diskretnih vrednosti funkcije na nekom skupu tačaka dobiti neprekidnu funkciju koja je definisana u svakoj tački, odnosno audio signal koji je moguće slušati. Problem transformisanja digitalne informacije u neprekidni signal nazivamo problemom *rekonstrukcije* (eng. reconstruction). Neki jednostavni načini da to uradimo jesu da konstruišemo deo-po-deo konstantnu funkciju kod koga je vrednost funkcije u nekoj tački jednaka vrednosti u najbližoj tački uzorka ili deo-po-deo linearnu funkciju koja vrši linearnu interpolaciju između vrednosti dva susedna uzorka. Međutim, ovakvi vidovi interpolacije mogu da ne uhvate sve karakteristike polazne funkcije, npr. lokalne ekstremume (slika 2.5).

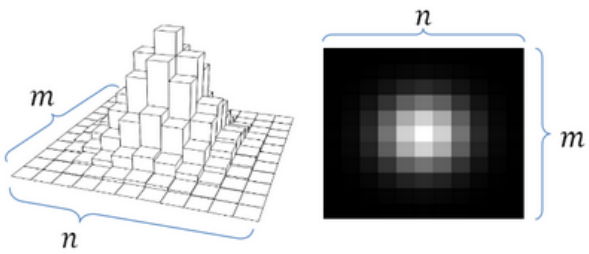


Slika 2.5: Razlika između polazne funkcije $f(x)$ i deo-po-deo linearne funkcije $\hat{f}(x)$ nad vrednostima uzoraka (preuzeto iz kursa Univerziteta Carnegie Mellone).

Kako bismo dobili bolju aproksimaciju polaznog signala, potrebno je gušće uzorkovati signal, odnosno povećati učestalost uzorkovanja. Primetimo da je to jedan od razloga zašto se audio signal uzorkuje 44100 puta u sekundi – kako ne bismo propustili nešto važno.

Mi smo navikli da na slike gledamo kao na niz piksela, odnosno kao diskretan signal. Međutim, svaka slika nastaje kao neprekidni signal – svetlost koja ulazi u pravu ili virtuelnu kameru je neprekidni signal, koji se onda uzorkuje procesom fotografisanja ili renderovanja. Na ovaj način dobijamo niz piksela, odnosno diskretni signal, tj. funkciju $\mathbb{Z}^2 \to \mathbb{R}$, koja vraća jednu vrednost za prosleđeni par celobrojnih koordinata (i, j) (slika 2.6). Vrednosti koje se čuvaju u diskretnim pikselima slike su uzorci originalnog signala. Polazni signal možemo rekonstruisati deo-po-deo konstantnom funkcijom (vrednost funkcije u tački odgovara vrednosti centra najbližeg piksela) ili deo-po-deo linearnom interpolacijom u svakom od smerova, što se naziva bilnearnom interpolacijom (o kojoj će biti više reči kasnije).

Vratimo se problemu rasterizacije trougla. Potrebno je uzorkovati *funkciju prekrivenosti trouglom* (eng. triangle coverage function) na mreži piksela. Ova funkcija je definisana za svaku tačku u ravni i vraća

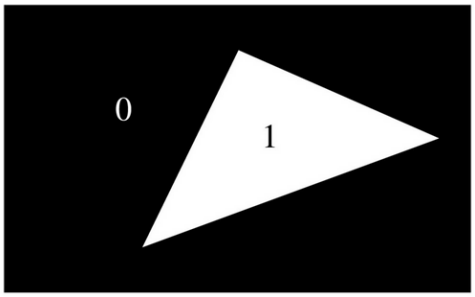


Slika 2.6: Diskretna reprezentacija slike data u vidu histograma koji označava količinu svetlosti po pikselu – na slici bela boja označava velike vrednosti, a crna male vrednosti.

jednu od vrednosti 1 i 0 (slika 2.7). Preciznije:

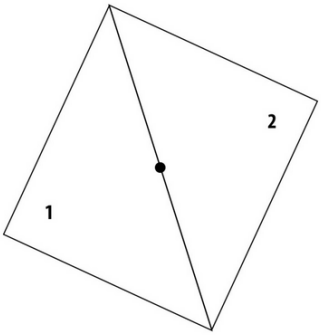
$$prekrivenost(x, y) = \begin{cases} 1, & \text{trougao sadrži tačku } (x, y) \\ 0, & \text{inače} \end{cases}$$

U najjednostavnijoj varijanti uzorkovanje vršimo u tačkama koje predstavljaju centre piksela. S obzirom da je centar piksela sa slike 2.4 pripada trouglu 3, a ne pripada trouglu 2, zaključili bismo da trougao 3 prekriva dati piksel, dok ga trougao 2 ne prekriva.



Slika 2.7: Vrednosti funkcije prekrivenosti trouglom (preuzeto iz kursa Univerziteta Carnegie Mellone).

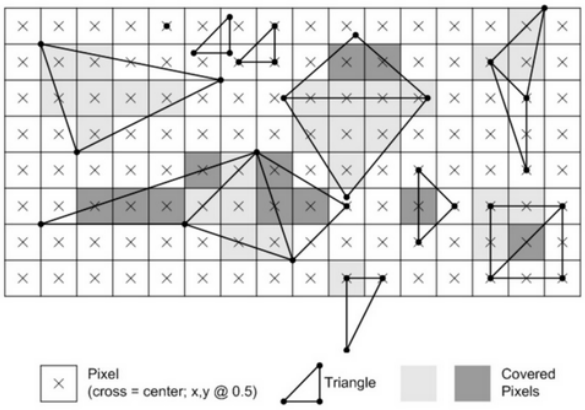
Razmotrimo slučaj kada dva trougla dele zajedničku stranicu (što će se neretko dešavati u praksi) i kada tačka u kojoj vršimo uzorkovanje baš pripada toj stranici (slika 2.8). Postavlja se pitanje da li u ovoj situaciji treba zaključiti da je tačka u kojoj vršimo uzorkovanje prekrivena prvim, drugim, sa oba trougla ili nijednim od njih. Iako ovo može delovati kao specijalan slučaj kome možda nije neophodno posvetiti previše vremena, važno je da svi koraci u zaključivanju budu dobro definisani. Naime, važno je da prikaz ne zavisi od nekih slučajnih izbora, npr. od redosleda u kome se ovi trouglovi razmatraju, već da garantujemo da ćemo kao izlaz dobiti uvek isti, korektni rezultat.



Slika 2.8: Situacija kada tačka u kojoj vršimo uzorkovanje pripada zajedničkoj stranici dva trougla.

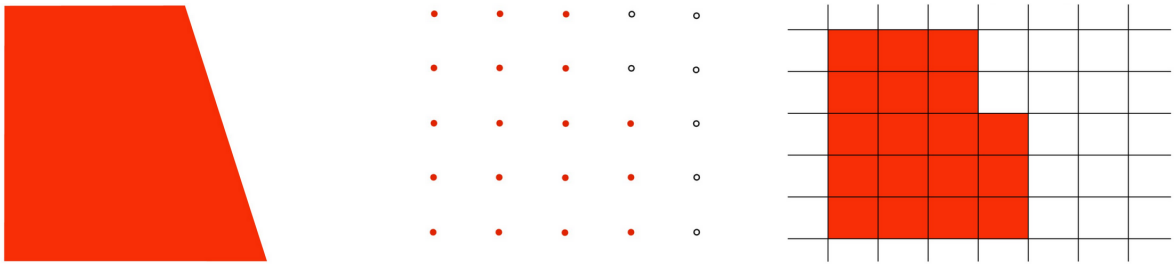
Pravilo koje koriste OpenGL-u i Direct3D jeste da kada se tačka uzorkovanja nalazi na samoj ivici trougla, klasifikuje se kao unutrašnja ako je u pitanju gornja ivica ili leva ivica trougla, pri čemu ivicu klasifikujemo kao gornju ako je horizontalna i nalazi se iznad ostalih ivica trougla, a kao levu ako nije

horizontalna i ceo trougao se nalazi desno od nje (primetimo da u opštem slučaju trougao može imati i dve leve ivice). U slučaju trouglova ilustrovanih na slici 2.8 za trougao 1 bismo zaključili da ne prekriva tačku uzorkovanja (jer je u pitanju desna stranica trougla 1), a za trougao 2 da je prekriva (jer je u pitanju leva stranica trougla 2). Razmotrimo sada različite scenarije ilustrovane na slici 2.9: u slučaju dva trougla koja dele horizontalnu ivicu koja sadrži tačke uzorkovanja (odnosno centre odgovarajućih piksela) od kojih je gornji potrebno obojiti tamno sivom bojom a donji svetlo sivom, zaključujemo da su ti pikseli prekriveni donjim trouglom jer je za taj trougao ivica klasifikovana kao gornja. Slično, ako razmotrimo trouglove u donjem desnom uglu slike koji dele kosu stranicu koja sadrži centar piksela – vidimo da je dati piksel obojen bojom donjeg trougla jer je za taj trougao to leva ivica trougla.



Slika 2.9: Primer većeg broja trouglova koji dele stranice – trouglovi su obojeni različitim nijansama sive kako bi se uočilo koji pikseli su klasifikovani da pripadaju kojim trouglovima

Pretpostavimo da smo funkciju prekrivenosti trouglom uzorkovali u centrima piksela za trougao prikazan na slici 2.10 levo. Vrednosti funkcije prekrivenosti u datim tačkama ilustrovane su na slici 2.10 u sredini. Kada bismo ove informacije pošlali uređaju za prikaz, dobili bismo prikaz poput onog na slici 2.10 desno.

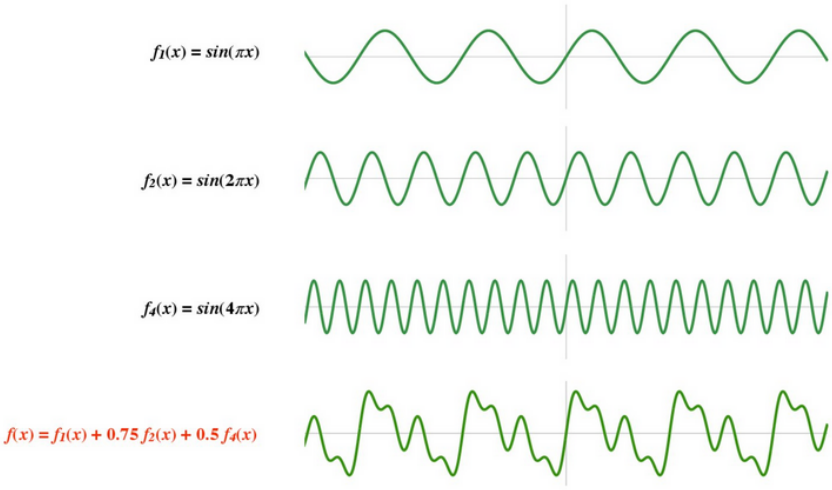


Slika 2.10: Aproksimacija prikaza uzorkovane slike trougla na ekranu (preuzeto iz kursa Univerziteta Carnegie Mellone).

Primetno je da se dobijeni prikaz rasterizovanog trougla razlikuje od polaznog i nije vizuelno prihvatljiv. U opštem slučaju ako imamo neprekidni signal koji želimo da zapamtimo u digitalnom formatu, mi vršimo njegovo uzorkovanje i pamtimo listu dobijenih vrednosti. Ako naknadno želimo da izvršimo rekonstrukciju polaznog signala na osnovu zapamćenih vrednosti često se dešava da se rekonstruisani signal razlikuje od polaznog i ovaj efekat naziva se *aliasing*. Da bismo razumeli zašto se ovo događa napravićemo kratak izlet u obradu signala.

2.2 Aliasing efekat i Najkvist-Šenonova teorema

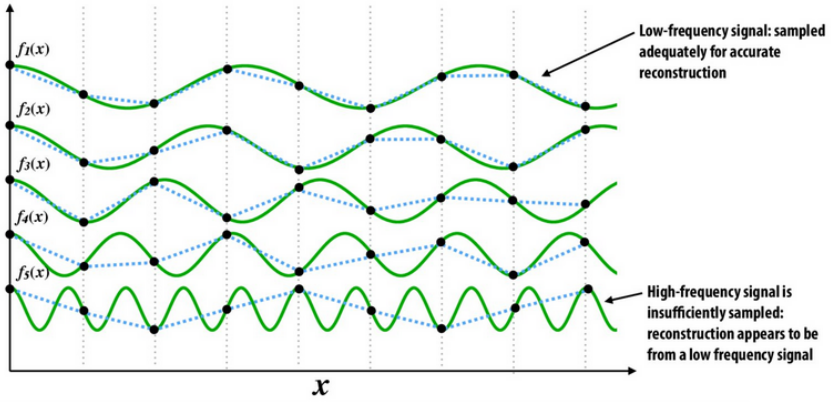
Signali su neprekidne funkcije koje često mogu delovati jako složeno. Međutim, važi da se proizvoljni signal može dekomponovati na nekoliko jednostavnijih funkcija (sinus i kosinus) različitih frekvencija, faze i amplitude. Specijalno, jednodimenzioni signal može se izraziti kao superpozicija (suma) sinusnih talasa određene frekvencije. Pritom, više frekvencije odgovaraju brzoj promeni, a niže frekvencije sporijoj promeni vrednosti signala. Na slici 2.11 je ilustrovan jedan signal $f(x)$ koji se može predstaviti kao suma tri različite sinusne funkcije: $f(x) = f_1(x) + 0.75 \cdot f_2(x) + 0.5 \cdot f_4(x)$.



Slika 2.11: Razlaganje 1D signala na različite frekvencije (preuzeto iz kursa Univerziteta Carnegie Mellone).

Kao primer jednodimenzionog signala razmotrimo ponovo slučaj audio signala: sinusni talas neke fiksne frekvencije predstavlja konstantan ton; pritom što je frekvencija veća to je ton viši. Kakav bismo efekat dobili kada bismo frekvenciju sinusnog talasa konstantno uvećavali tokom vremena? Očekivali bismo da i visina tona konstantno raste što se i dešava na samom početku, međutim, nakon određenog vremena, visina tona počinje da opada, pa onda opet da raste itd. Postavlja se pitanje zašto se ovo dešava. Pokazuje se da ako pokušamo da rekonstruišemo nedovoljno često uzorkovan signal visoke frekvencije kao rezultat dobijamo signal niske frekvencije. Kao što smo već pomenuli u situaciji kada rekonstruisani signal daje pogrešnu sliku o tome kako je izgledao polazni signal kažemo da je došlo do aliasinga.

Razmotrimo pet različitih funkcija ilustrovanih na slici 2.12: funkcija $f_1(x)$ sporo osciluje, dok funkcija $f_5(x)$ jako brzo osciluje. Svaka od funkcija je uzorkovana na istom skupu tačaka i nakon toga je izvršena rekonstrukcija polaznog signala deo-po-deo linearnom aproksimacijom. U slučaju funkcije $f_1(x)$ koja ima najmanju frekvenciju dobijeni rezultat je sličan polaznom signalu. Međutim, ako razmotrimo funkciju $f_5(x)$ koja ima jako visoku frekvenciju na ovaj način dobijamo jako lošu aproksimaciju i na osnovu rekonstruisane funkcije bismo zaključili da je polazni signal imao nisku frekvenciju.

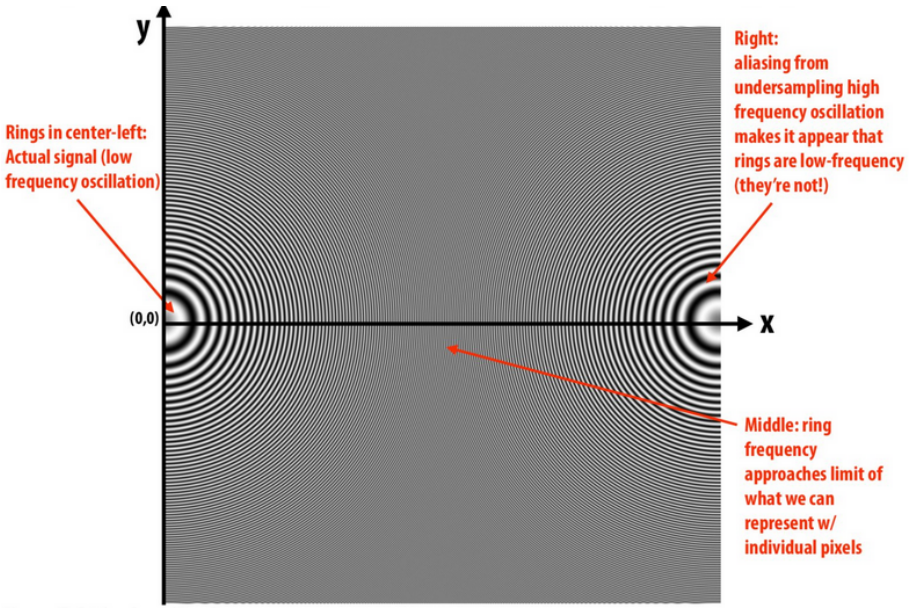


Slika 2.12: Deo-po-deo linearna aproksimacija sinusnih funkcija različitih frekvencija na osnovu uzorkovanja na istom skupu tačaka (preuzeto iz kursa Univerziteta Carnegie Mellone).

Sinusne talase je moguće uočiti i u dvodimenzionim signalima, poput slika, gde maksimalnu pozitivnu amplitudu označavamo belom, a maksimalnu negativnu amplitudu crnom bojom. Slike je, takođe, moguće dekomponovati na sumu različitih frekvencija. Naime, ako bismo iz slike izolovali samo niske frekvencije, dobili bismo glatke prelaze između regiona različitih intenziteta i efekat zamućene slike. Ako bismo izolovali frekvencije srednjeg opsega, dobili bismo one delove slike kod kojih gradijent nije gladak, a ako izolujemo samo visoke frekvencije izolovaćemo ivice na slici. I u slučaju rada sa slikama i

rekonstrukcije signala na osnovu uzorkovanih vrednosti, moguće je da dođe do aliasing efekta, ukoliko slika sadrži visoke frekvencije a imamo grubu mrežu piksela.

Zanimljiv je i naredni, namenski kreiran primer: razmotrimo ponašanje funkcije $\sin(x^2+y^2)$ prikazano na slici 2.13. Kako se udaljavamo od koordinatnog početka, funkcija počinje sve brže da osciluje, odnosno prstenovi postaju sve tanji. Međutim, primetimo da se na desnom kraju slike ponovo javljaju prstenovi niske frekvencije. Zašto se ovo dešava? Ovi prstenovi nisu tu u realnosti, već je razlog ovakvog efekta to što smo se toliko udaljili od onoga što možemo da predstavimo na slici da dobijamo aliasing efekat. Naime, funkcija jako brzo osciluje i na datom skupu tačaka u kojima vršimo uzorkovanje nije moguće uhvatiti sve njene karakteristike.



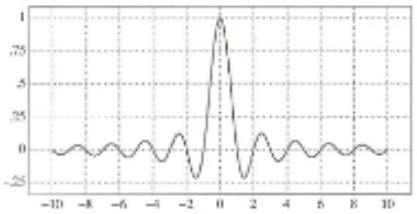
Slika 2.13: Primer funkcije koja jako osciluje (preuzeto iz kursa Univerziteta Carnegie Mellone).

Interesantna je i sledeća pojava koja se može nekada uočiti na video-snimcima: kada se auto kreće i konstantno ubrzava, mi uočavamo kako se točkovi automobila sve brže kreću unapred, a onda odjednom deluje kao da su točkovi statični ili se kreću unazad. Ovaj efekat je posledica toga što video zapis prikazuje 30 slika u sekundi, te se prilikom prebrzog okretanja točka uzorkuje neki slučajan momenat pri okretanju točka čime dobijamo neočekivano ponašanje. Ovaj efekat poznat je pod nazivom temporalni aliasing.

Postavlja se pitanje u kojim tačno situacijama dolazi do aliasinga. Odgovor na ovo pitanje daje nam Najvkist-Šenonova teorema koja glasi: ako signal nema frekvencije više od ω_0, on se može savršeno rekonstruisati ako se uzorkuje sa učestanošću dvostruko većom od najviše frekvencije ω_0. Problem sa primenom ove teoreme je to što signali često nisu ograničenog opsega – npr. neku ivicu na slici je teško izraziti kao sinusoidnu funkciju: morali bismo da dodamo beskonačnu seriju sve većih frekvencija, kako bismo dobili deo-po-deo konstantnu funkciju. Iz tog razloga se nekada koristi prefiltriranje, kojim se ograničava maksimalna frekvencija prisutna u signalu.

Za interpolaciju se umesto deo-po-deo konstantne ili deo-po-deo linearne interpolacije koristi tzv. $sinc$ funkcija (slika 2.14):

$$sinc(x) = \frac{1}{\pi x}\sin(\pi x)$$



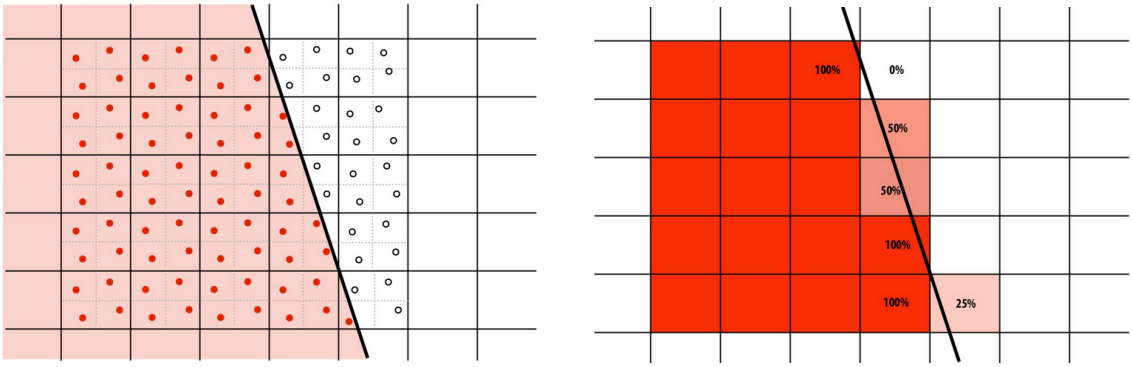
Slika 2.14: Funkcija $sinc$.

Pokazuje se da se ovom vrstom interpolacije dobija mnogo bolja aproksimacija polaznog signala. Ono što je nezgodno pri korišćenju funkcije $sinc$ je to što ona ima beskonačan opseg i utiče na sve piksele na slici.

Ukoliko se signal uzorkuje sa učestanošću manjom od dvostruke maksimalne frekvencije prisutne u signalu, dolazi do neželjenih efekata na slikama, poput nazubljenih linija. Drugi mogući efekat je sledeći: razmotrimo primer slike beskonačne crno-bele šahovske table koja se udaljava od posmatrača iznad koje je crna pozadina (slika 2.16 levo). Boja svakog od piksela na slici je crna ili bela. Možemo primeti da slika izgleda neprirodno, odnosno da se na horizontu, tj. u delu slike koji ima visoku frekvenciju javlja tzv. Moire uzorak (eng. Moiré pattern). Aliasing efekat se ne može u potpunosti eliminisati jer svaka uzorkovana reprezentacija ne uspeva da uhvati dovoljno visoke frekvencije prisutne u signalu, ali je ovaj efekat moguće donekle ublažiti.

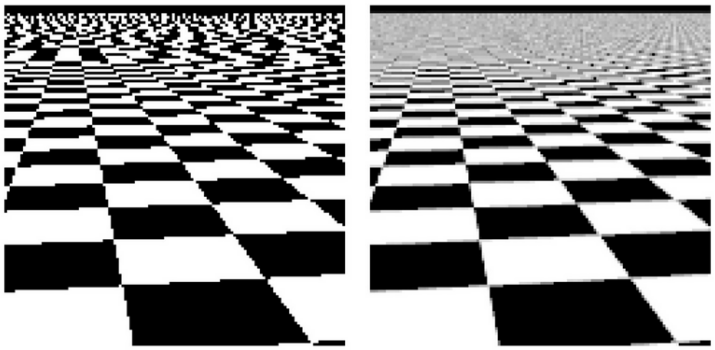
2.3 Nadsemplovanje

Vratimo se problemu rasterizacije trougla: za svaki piksel slike potrebno je utvrditi koliki je njegov procenat prekriven trouglom. Do sada smo uzorkovanje vršili po jednom za svaki piksel i to u tački koja predstavlja centar piksela. Na osnovu prethodnog razmatranja zaključujemo da je potrebno povećati frekvenciju uzorkovanja, odnosno funkciju prekrivenosti uzorkovati veći broj puta za svaki od piksela (slika 2.15 levo). Ova antialiasing tehnika naziva se *nadsemplovanje* (eng. supersampling). Na ovaj način kao da se povećava rezolucija na kojoj se vrše izračunavanja, međutim, treba da budemo svesni da je na kraju ipak potrebno prikazati sliku na polaznoj mreži piksela.



Slika 2.15: Uzorkovanje funkcije prekrivenosti na većem broju uzoraka po pikselu – nadsemplovanje (preuzeto iz kursa Univerziteta Carnegie Mellone).

Nakon uzorkovanja funkcije prekrivenosti trouglom na većem broju tačaka po pikselu, potrebno je za svaki od piksela uprosečiti dobijene vrednosti uzoraka iz tog piksela, odnosno izračunati procenat broja uzoraka po pikselu koji su prekriveni (slika 2.15 desno). Ovo odgovara primeni tzv. *filtera kutije* (eng. box filter) kojim se vrši uprosečavanje dobijenih vrednosti.



Slika 2.16: Slika šahovske table bez nadsemplovanja i sa primenom nadsemplovanja sa 1024 uzoraka po pikselu.

Na slici 2.16 ilustrovano je poboljšanje prikaza dobijeno primenom nadsemplovanja korišćenjem $32 \times 32 = 1024$ uzoraka po pikselu. S obzirom da ova slika sadrži visoke frekvencije jer ima nagle

prelaze iz crnih u bela polja, na osnovu Najkvist-Šenonove teoreme možemo zaključiti da ni korišćenjem ovolikog broja uzoraka po pikselu nećemo dobiti savršen prikaz.

2.4 Utvrđivanje da li je tačka u trouglu

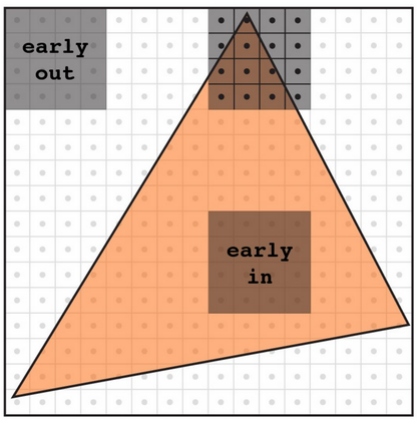
U svim prethodnim razmatranjima smo problem ispitivanja da li se neka tačka nalazi unutar ili van trougla razmatrali kao osnovnu primitivu koju znamo kako da realizujemo. Za dati trougao $P_0P_1P_2$ i tačku Q važi da se tačka Q nalazi unutar trougla $P_0P_1P_2$ ako i samo ako je orijentacija trouglova $\triangle P_0P_1Q$, $\triangle P_1P_2Q$ i $\triangle P_2P_0Q$ ista. Dakle, ispitivanje pripadnosti tačke trouglu svodi se na tri operacije računanje orijentacije trojke tačaka.

Prilikom rasterizacije trougla potrebno je za sve piksele slike ispitati da li se nalaze u trouglu. Međutim, trougao može zauzimati jako mali procenat čitave slike, te nije efikasno eksplicitno proveravati sve piksele slike. Bolji pristup bio bi izračunati granični opseg trougla (eng. bounding box) i za svaki piksel ovog manjeg pravougaonika utvrditi da li se nalazi u trouglu ili ne.

Najjednostavniji pristup rasterizaciji trougla podrazumeva da sistematično prolazimo kroz piksele i utvrđujemo koji pikseli su unutar trougla. Pošto očekujemo da za bliske piksele dobijemo slične vrednosti, ima smisla prolaziti redom kroz piksele, vrstu po vrstu i izračunavanja za piksele iz naredne vrste vršiti inkrementalno na osnovu vrednosti za piksele iz prethodne vrste. Ovakav, inkrementalni pristup rasterizaciji trougla pretpostavlja serijsko izvršavanje, dok moderni grafički hardver ima paralelnu strukturu.

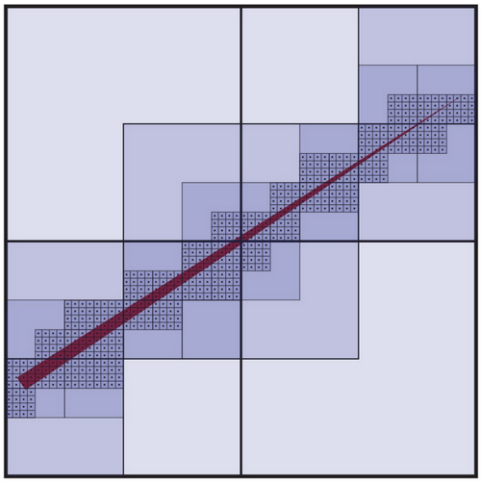
Umesto serijskog izvršavanja možemo paralelno testirati sve uzorke iz graničnog opsega trougla na pripadnost trouglu. Pokazuje se da mogućnost paralelnog izvršavanja operacija nadomešćuje cenu većeg broja testiranja uzoraka koje je potrebno izvršiti. Primetimo da u slučaju kada je trougao dugačak i tanak i koso postavljen, njegov granični opseg biće veliki, te veliki broj uzoraka treba testirati na pripadnost trouglu, a jako mali broj piksela će efektivno pripadati trouglu. Dakle, unutar graničnog opsega postojaće velike oblasti piksela koje bismo nekako mogli unapred eliminisati iz daljeg ispitivanja.

Paralalno izvršavanje testova je u ovom scenariju moguće unaprediti tako što granični opseg trougla podelimo na blokove srednje veličine i pre paralelnog testiranja pojedinačnih piksela na pripadnost trouglu, za svaki blok možemo da proverimo da li ima presek sa trouglom: ako blok ne preseca trougao, eliminišemo sve piksele iz bloka, ako je čitav blok sadržan unutar trougla, zaključujemo da trougao prekriva sve tačke tog bloka. Ako ne važi ništa od prethodnog, testiramo u paraleli pojedinačno sve tačke iz bloka (slika 2.17).



Slika 2.17: Hijerarhijski pristup rasterizaciji trougla (preuzeto iz kursa Univerziteta Carnegie Mellone).

Postavlja se pitanje koja je optimalna veličina bloka. To zavisi od konkretnog trougla. Stoga, da bismo dobili idealnu veličinu bloka i da bismo brzo eliminisali velike blokove van/unutar trougla, koristimo hibridni pristup rasterizaciji trougla. Naime, umesto da unapred fiksiramo neku veličinu bloka, idemo od većih ka manjim blokovima, počev od podele graničnog opsega na 4 jednaka dela, i testiramo da li trougao ima presek sa svakim od ovih blokova; svaki blok koji ima presek sa trouglom delimo na četiri manja bloka i ponavljamo testove za njih. Na ovaj način preskačemo one blokove koje trougao ne seče, dok ostale blokove delimo na još manje (slika 2.18). U nekom trenutku prestajemo sa daljim podelama jer postaje efikasnije testirati paralelno sve piksele iz bloka.



Slika 2.18: Hibridni pristup rasterizaciji trougla (preuzeto iz kursa Univerziteta Carnegie Mellone).

2.5 Pitanja

2.1 Šta je rasterizacija?

2.2 Kako bi izgledala vektorska, a kako rasterska slika dva pravougaonika sa stranicama paralelnim koordinatnim osama? Koja od ove dve slike bi zauzimala manje memorije ako je rasterska slika dimenzije 256×256 piksela? Zašto?

2.3 Navesti primer gde se u praksi koristi vektorska grafika, a gde rasterska.

Glava 3

Geometrijske transformacije

Geometrijske transformacija je funkcija koja preslikava skup tačaka u skup tačaka (pri čemu su dati skupovi najčešće $\mathbb{R}^2$ ili $\mathbb{R}^3$). Geometrijske transformacije igraju važnu ulogu u oblasti računarske grafike: one omogućavaju pravljenje složenih modela postavljanjem dimenzija i pozicioniranjem njihovih sastavnih delova; uz pomoć njih moguće je izvršiti preslikavanje iz jednog u drugi koordinatni sistem, na primer iz koordinatnog sistema objekta u svetski koordinatni sistem; one omogućavaju postavljanje virtuelne kamere na sceni i sagledavanje scene iz različitih tačaka. Geometrijske transformacije čine i esencijalni deo računarski generisanih animacija jer omogućavaju definisanje transformacija koje deluju na objekat na sceni tokom vremena, kako bi se stekao osećaj kretanja. Međutim, geometrijske transformacije se koriste i na mnogim drugim mestima, kao što je projektovanje trodimenzionih objekata u ravan, preslikavanje dvodimenzionih tekstura na trodimenzione objekte, za projektovanje senki trodimenzionih objekata na druge objekte itd.

U ovom poglavlju podsetićemo se osnovnih geometrijskih transformacija u ravni i trodimenzionom prostoru i razmotriti neke od karakterističnih problema koji se javljaju u računarskoj grafici i u kojima geometrijske transformacije igraju važnu ulogu, kao što je promena koordinatnog sistema.

3.1 2D transformacije

Najpre ćemo razmatrati osnovne geometrijske transformacije u ravni, a nakon toga razmatraćemo i odgovarajuće geometrijske transformacije u trodimenzionom prostoru. Translacija, skaliranje, rotacija i smicanje, koje će biti detaljno diskutovane u nastavku teksta, predstavljaju esencijalni deo velikog broja grafičkih aplikacija.

3.1.1 Linearne transformacije

Linearne transformacije zbog svojih pogodnih osobina igraju značajnu ulogu u računarskoj grafici. Podsetimo se njihove definicije. Za funkciju f kažemo da je *linearna transformacija* ako važi:

- $f(v + w) = f(v) + f(w)$ za sve vrednosti v i w iz domena funkcije f;
- $f(cv) = cf(v)$ za sve skalare c i vrednosti v iz domena funkcije f.

Geometrijski posmatrano, linearne transformacije predstavljaju transformacije tačaka kod kojih se koordinatni početak preslikava sam u sebe i kod kojih se čuva kolinearnost tačaka (prave se slikaju u prave). Pored toga, linearne transformacije čuvaju kolinearnost tačaka i paralelnost pravih: međusobno paralelne prave slikaju se u paralelne prave. Nama će pored razmatranja transformacija u terminima matrica kojima su one zadate, biti jednako interesantna i karakterizacija transformacija u terminima njenih *invarijanti*, odnosno svojstava koje taj tip transformacija čuva. Naime, pokazuje se da ljudi mogu i bez poznavanja formula i matrica transformacija lako prepoznati tip transformacije uočavanjem veličina i svojstava koje transformacija čuva.

Primeri linearnih transformacija su skaliranje u odnosu na koordinatni početak, rotacija oko koordinatnog početka, smicanje u pravcu neke od koordinatnih osa, ali ne i translacija jer se njome koordinatni početak izmešta iz svog položaja.

Linearne transformacije u ravni mogu se predstaviti kao množenje matricom dimenzije 2×2. Naime, ako sa $v = [x\ y]^T$ označimo koordinate proizvoljne tačke ravni i ako se ona pri linearnoj transformaciji

T slika u tačku $v' = [x' \;\; y']^T$, onda važi veza

$$v' = M \cdot v$$

pri čemu je sa M označena matrica linearne transformacije T koja je dimenzije 2×2. Slično, linearnu transformaciju u trodimenzionom prostoru možemo predstaviti kao množenje matricom dimenzije 3×3.

Podsetimo se da se *baza* dvodimenzionog vektorskog prostora definiše kao skup vektora $\{\vec{V_1}, \vec{V_2}\}$ za koje važi:

- vektori $\vec{V_1}$ i $\vec{V_2}$ su linearno nezavisni,
- proizvoljni vektor $\vec{V}$ iz datog vektorskog prostora može se izraziti kao linearna kombinacija vektora iz ovog skupa: $\vec{V} = c_1\vec{V_1} + c_2\vec{V_2}$.

Vektore iz ovog skupa nazivamo *baznim vektorima*.

Neka je $\vec{e_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ i $\vec{e_2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ standardna baza vektorskog prostora dimenzije 2 i neka je $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ matrica linearne transformacije T. Bazni vektori $\vec{e_1}$ i $\vec{e_2}$ se pri ovoj transformaciji slikaju u vektore:

$$T(\vec{e_1}) = T(\begin{bmatrix} 1 \\ 0 \end{bmatrix}) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} a \\ c \end{bmatrix}$$

$$T(\vec{e_2}) = T(\begin{bmatrix} 0 \\ 1 \end{bmatrix}) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix}$$

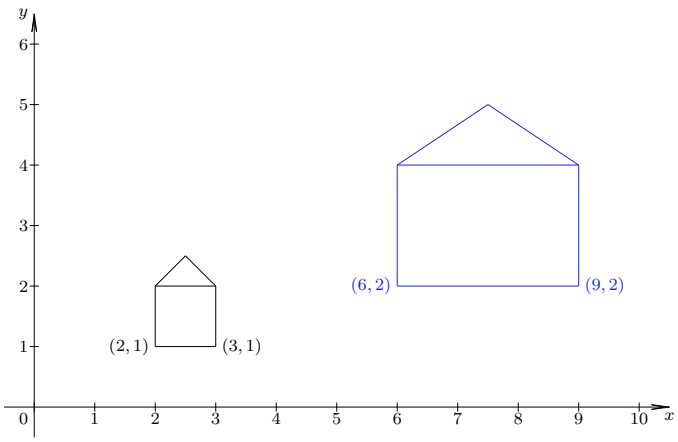
Primetimo da je prva kolona matrice transformacije T jednaka slici prvog baznog vektora, a druga kolona slici drugog baznog vektora. Ovo nam daje jednu moguću strategiju za izvođenje matrica linearnih transformacija: kolonu po kolonu matrice transformacije izračunavamo razmatranjem na koji način data transformacija deluje na bazne vektore standardne baze.

3.1.2 Osnovne geometrijske transformacije

Skaliranje

Transformacija promene veličine objekta naziva se *skaliranje* (*istezanje*). U daljem tekstu, kada ne bude bilo eksplicitno naglašena tačka u odnosu na koju se vrši skaliranje, podrazumevaćemo skaliranje u odnosu na koordinatni početak.

Tačke u ravni se mogu skalirati duž x i duž y ose i pritom nije neophodno da faktori skaliranja po x i po y osi budu jednaki. Na primer, dozvoljeno je vršiti skaliranje u odnosu na koordinatni početak po x osi za faktor 3, a po y osi za faktor 2 ($s_x = 3, s_y = 2$). Ovo podrazumeva da se x koordinata svake tačke povećava tri puta, a y koordinata dva puta (slika 3.1).



Slika 3.1: Skaliranje sa faktorima $s_x = 3$ i $s_y = 2$.

Izvedimo matricu skaliranja S razmatranjem na koji način vektori $\vec{e_1}$ i $\vec{e_2}$ treba da budu transformisani: jedinični vektor $\vec{e_1}$ u smeru x koordinatne ose treba da se slika u vektor $s_x \cdot \vec{e_1}$ i, slično, jedinični vektor $\vec{e_2}$ u smeru y koordinatne ose treba da se slika u vektor $s_y \cdot \vec{e_2}$.

$$\vec{e_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \rightarrow s_x \cdot \vec{e_1} = \begin{bmatrix} s_x \\ 0 \end{bmatrix}$$

$$\vec{e_2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \rightarrow s_y \cdot \vec{e_2} = \begin{bmatrix} 0 \\ s_y \end{bmatrix}$$

Na ovaj način izveli smo kolone matrice skaliranja S, te je matrica skaliranja u odnosu na koordinatni početak jednaka:

$$S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

odnosno, transformaciju skaliranja u odnosu na koordinatni početak možemo zapisati matrično kao:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix},$$

odnosno skraćeno:

$$v' = S \cdot v$$

Primetimo da je matrica S dijagonalna matrica, odnosno pri ovoj transformaciji svaku od koordinata slike dobijamo isključivo na osnovu te iste koordinate originalne tačke, množenjem odgovarajućim faktorom skaliranja:

$$x' = s_x \cdot x$$

$$y' = s_y \cdot y$$

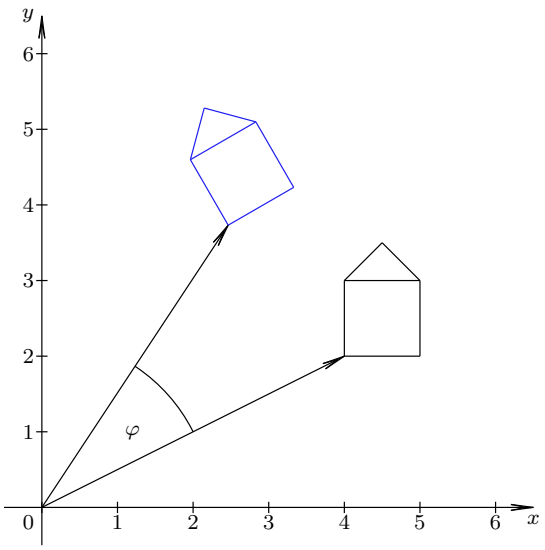
Skaliranjem u odnosu na koordinatni početak se ne čuvaju uglovi između pravih u ravni (osim kada je skaliranje uniformno, tj. kada važi $s_x = s_y$). Takođe, ako objekat ne počinje u koordinatnom početku skaliranje će ga približiti ili udaljiti od koordinatnog početka (što je često u suprotnosti sa onim što korisnik očekuje).

Primer 3.1. Na slici 3.1 prikazano je kako se skaliranjem sa različitim faktorima u smeru x i y ose dobija objekat kod koga nisu očuvane veličine uglova niti odnos dužina stranica. Takođe, slika objekta je udaljenija od koordinatnog početka od polaznog objekta jer objekat ne počinje u koordinatnom početku a koeficijenti skaliranja su po apsolutnoj vrednosti veći od 1. Ukoliko bismo želeli da kao sliku dobijemo objekat koji počinje u istoj tački, odnosno čije se donje levo teme poklapa sa donjim levim temenom polazne figure, bilo bi potrebno izvršiti skaliranje u odnosu na tačku sa koordinatama $(2, 1)$. Matricu skaliranja u odnosu na proizvoljnu tačku različitu od koordinatnog početka izvešćemo u nekom od narednih poglavlja.

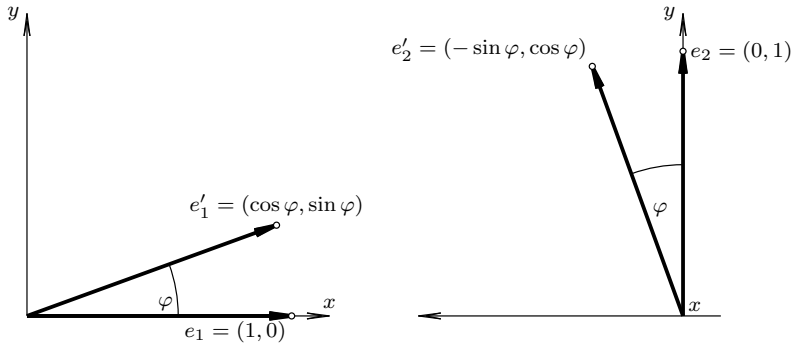
Rotacija

Rotacija u ravni se zadaje svojim centrom i uglom za koji treba izvršiti rotaciju. Razmotrimo rotaciju oko koordinatnog početka za ugao φ, pri čemu su uglovi pozitivno orijentisani (slika 3.2). Izvedimo matricu rotacije R razmatranjem na koji način ona transformiše vektore $\vec{e_1}$ i $\vec{e_2}$ (slika 3.3). Vektori $\vec{e_1}$ i $\vec{e_2}$ slikaju se u vektore dužine 1, dok se njihove x i y koordinate dobijaju iz pravouglih trouglova čiji je jedan ugao φ (obratiti pažnju da je x koordinata vektora $\vec{e_1}'$ negativna).

- vektor $\vec{e_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ slika se u vektor $\vec{e_1'} = \begin{bmatrix} \cos\varphi \\ \sin\varphi \end{bmatrix}$
- vektor $\vec{e_2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ slika se u vektor $\vec{e_2'} = \begin{bmatrix} -\sin\varphi \\ \cos\varphi \end{bmatrix}$



Slika 3.2: Rotacija za ugao φ.



Slika 3.3: Delovanje rotacije za ugao φ na jedinične vektore u smeru koordinatnih osa.

Stoga je matrica rotacije oko koordinatnog početka za ugao φ jednaka

$$R = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}$$

tj. transformacija koordinata tačke rotacijom može se opisati narednom matričnom jednačinom:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

ili skraćeno:

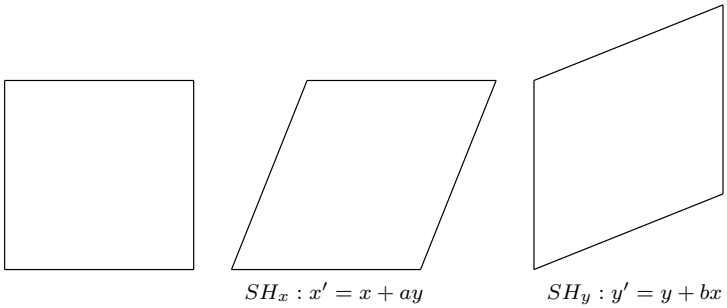
$$v' = R \cdot v$$

Odgovarajuće jednačine rotacije glase:

$$x' = x \cdot \cos\varphi - y \cdot \sin\varphi$$

$$y' = x \cdot \sin\varphi + y \cdot \cos\varphi$$

Rotacijom se čuvaju dužine duži, kao i uglovi među parovima pravih. Specijalno, paralelne prave se rotacijom preslikavaju u paralelne prave.



Slika 3.4: Smicanje u pravcu x i y koordinatne ose.

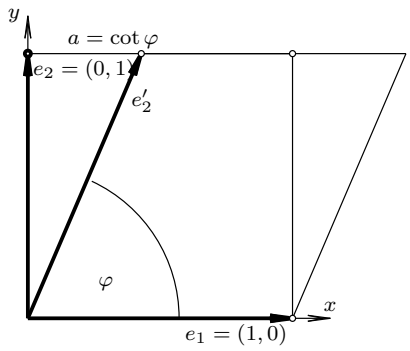
Smicanje

Transformacija *smicanja* ili *iskošenja* (eng. shear) podrazumeva da objekat treba iskositi u pravcu neke od koordinatnih osa, dok pritom vrednost druge koordinate ostaje ista. Na ovaj način, recimo, kvadrat postaje paralelogram (slika 3.4).

Izvedimo matricu smicanja SH_x u pravcu x ose razmatranjem na koji način ona treba da utiče na vektore $\vec{e_1}$ i $\vec{e_2}$ (slika 3.5): vektor $\vec{e_1}$ slika se u samog sebe, dok se vektor $\vec{e_2}$ slika u vektor čija je y koordinata jednaka 1, a x koordinata se dobija iz uslova $1/x = \tan\varphi$, tj. $x = \cot\varphi$.

- prvi bazni vektor $\vec{e_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ slika se u vektor $\vec{e_1'} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$
- drugi bazni vektor $\vec{e_2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ slika se u vektor $\vec{e_2'} = \begin{bmatrix} \cot\varphi \\ 1 \end{bmatrix}$

gde je sa φ označen ugao za koji se vrši iskošenje. Vrednost $\cot\varphi$ označićemo sa a i u daljem tekstu zvaćemo je *koeficijent smicanja*.



Slika 3.5: Delovanje smicanja u pravcu x ose na jedinične vektore u smeru koordinatnih osa.

Dakle, matrica smicanja u pravcu x ose jednaka je:

$$SH_x = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}$$

a transformacija smicanja u pravcu x ose može se opisati na sledeći način:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

odnosno skraćeno:

$$v' = SH_x \cdot v$$

Na sličan način bi se izvela i matrica smicanja u pravcu y ose sa koeficijentom smicanja b:

$$SH_y = \begin{bmatrix} 1 & 0 \\ b & 1 \end{bmatrix}$$

odnosno smicanje u pravcu y ose zadaje se jednačinom:

$$v' = SH_y \cdot v$$

Jednačine smicanja u pravcu x koordinatne ose glase:

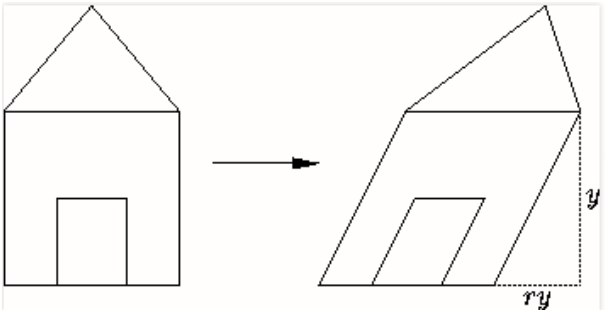
$$x' = x + a \cdot y$$

$$y' = y$$

a smicanja u pravcu y koordinatne ose:

$$x' = x$$

$$y' = y + b \cdot x$$



Slika 3.6: Smicanje objekta sa koeficijentom $r = 1/2$ u pravcu x koordinatne ose.

Primetimo da prilikom izvođenja ove transformacije vrednosti jedne koordinate ostaju fiksne (npr. prilikom smicanja u pravcu x koordinatne ose y koordinate tačaka se ne menjaju). Smicanjem se ne čuvaju dužine (osim na pravcu u kome se vrši istezanje), niti se čuvaju uglovi (slika 3.6). Smicanjem se pak čuvaju površine geometrijskih figura.

Opšti oblik linearne transformacije

Razmotrimo opšti oblik matrice linearne transformacije $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$. Ukoliko je determinanta ove matrice $ad - bc$ jednaka nuli, onda ovu transformaciju zovemo *singularnom* i za nju ne postoji inverzna transformacija. Takav je recimo slučaj transformacije definisane matricom $\begin{bmatrix} 1 & -2 \\ 2 & -4 \end{bmatrix}$. Naime, ovakve transformacije slikaju kompletnu ravan u pravu i njom se gubi veza između tačaka iz domena preslikavanja sa tačkama iz kodomena preslikavanja: neke tačke iz kodomena ne odgovaraju nijednoj tački iz domena, dok neke tačke kodomena odgovaraju većem broju tačaka iz domena. Primetimo da su kolone matrica singularnih transformacija linearno zavisne.

Ukoliko je vrednost determinante $ad - bc$ date matrice različita od nula, matrica transformacije je invertibilna i inverz matrice jednak je:

$$\frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Vrednost determinante matrica svih pomenutih linearnih transformacija (skaliranja, rotacije i smicanja) različita je od nule i svaka od tih transformacija ima njoj inverznu. Mi u ovom materijalu nećemo računati matrice inverznih transformacija korišćenjem ove uopštene formule, već ćemo razmatrati matrice transformacija inverznih nekoj konkretnoj transformaciji, poput rotacije i skaliranja.

Translacija

Translacija nije linearna transformacija jer se njome koordinatni početak ne čuva, te se ona ne može predstaviti putem množenja matricom dimenzije 2×2. Međutim, translaciju je moguće predstaviti korišćenjem sabiranja:

$$x' = x + t_x \quad y' = y + t_y$$

odnosno u matričnoj formi:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

tj. skraćeno:

$$v' = v + T$$

Kompoziciji linearnih transformacija odgovara proizvod odgovarajućih matrica transformacija. Međutim, sabiranje sa vektorom za kombinovanje translacije sa linearnim transformacijama nije konzistentno sa množenjem matrica. Naime, poželjno je da sve pomenute transformacije: translacija ($v' = v + T$), skaliranje ($v' = S \cdot v$), rotacija ($v' = R \cdot v$) i smicanje ($v' = SH \cdot v$) budu predstavljene na isti način – putem množenja matrica, kako bismo kombinaciju proizvoljnog niza transformacija mogli da izrazimo kompozicijom njihovih matrica. Iz ovog razloga prelazimo na homogene koordinate.

3.1.3 Homogene koordinate

Nemački matematičar i astronom Avgust Mebius (August Mobius) je još davne 1827. godine uveo koncept homogenih koordinata kao pogodan sistem koordinata u projektivnoj geometriji. Homogene koordinate su našle brojne primene, a u računarskoj grafici se koriste od šezdesetih godina dvadesetog veka.

Korišćenjem homogenih koordinata tačka u ravni se predstavlja trojkom vrednosti (x, y, W), pri čemu bar jedna od vrednosti x, y i W nije jednaka 0. Dve trojke (x, y, W) i (x', y', W') predstavljaju istu tačku u ravni ako postoji broj t takav da je $x = tx'$, $y = ty'$ i $W = tW'$. Tako, na primer, trojke $(4, 3, 7)$, $(8, 6, 14)$ i $(4/7, 3/7, 1)$ predstavljaju istu tačku. Očigledno je da postoji beskonačno mnogo načina za predstavljanje jedne tačke u ravni njenim homogenim koordinatama.

Ako je vrednost $W \neq 0$, onda tačku možemo predstaviti u obliku $(x/W, y/W, 1)$ i ovaj proces nazivamo *homogenizacija*, dok $(x/W, y/W)$ nazivamo *Dekartovim koordinatama homogene tačke*. Tačku sa koordinatama $(x, y, 0)$ nazivamo *beskonačno dalekom tačkom* u pravcu (x, y).

Homogenim tačkama u ravni odgovaraju uređene trojke vrednosti, a te trojke odgovaraju tačkama u trodimenzionom Dekartovom prostoru. Skupu svih trojki kojima odgovara jedna homogena tačka odgovara prava Dekartovog prostora koja prolazi kroz koordinatni početak (slika 3.7). Zaista, sve tačke oblika (tx, ty, tW) za $t \neq 0$ pripadaju jednoj pravoj Dekartovog prostora. Ako izvršimo homogenizaciju koordinata tačke, dobijamo njenu reprezentaciju $(x/W, y/W, 1)$ koja u dekartovskom smislu pripada ravni $W = 1$. Dakle, sve homogenizovane tačke formiraju ravan definisanu jednačinom $W = 1$.

S obzirom na to da se tačke u ravni u homogenim koordinatama predstavljaju trojkama vrednosti, matrice transformacije u ravni u homogenim koordinatama su dimenzije 3×3.

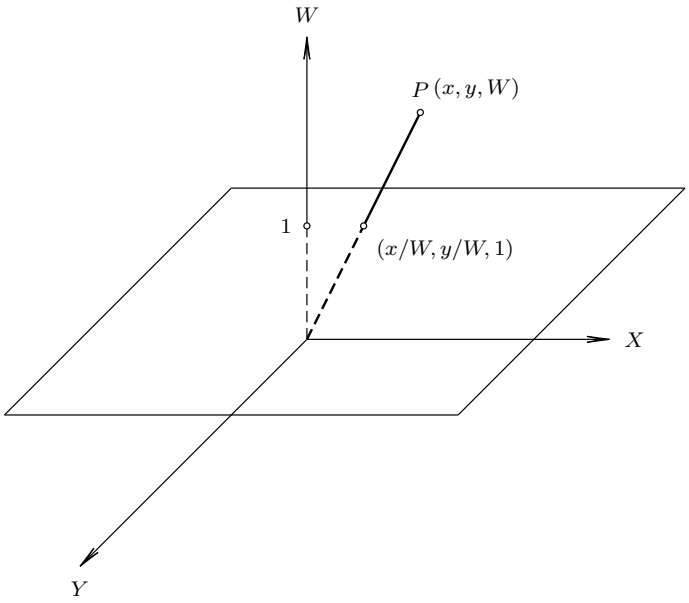
3.1.4 2D transformacije u homogenim koordinatama

Izrazimo sada prethodno pomenute geometrijske transformacije (skaliranje, rotaciju, smicanje i translaciju) u terminima homogenih koordinata. Kada budemo dobili homogene koordinate slike tačke, izvršićemo deljenje x i y koordinate vrednošću W koordinate i dobiti Dekartove koordinate slike tačke.

Tačku u ravni sa koordinatama $v = [x \;\; y]^T$ možemo predstaviti korišćenjem homogenih koordinata kao $v = [x \;\; y \;\; 1]^T$. Neka se tačka v preslikava u tačku sa homogenim koordinatama $v' = [x' \;\; y' \;\; 1]^T$ i neka je matrica transformacije u homogenim koordinatama jednaka:

$$\begin{bmatrix} m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 \\ m_7 & m_8 & m_9 \end{bmatrix}$$

Potrebno je da važi:



Slika 3.7: Skupu svih trojki koje predstavljaju homogene koordinate jedne tačke u ravni odgovara prava u trodimenzionom prostoru.

$$\begin{bmatrix} m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 \\ m_7 & m_8 & m_9 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

za svako x i y, odnosno potrebno je da važi $m_7x + m_8y + m_9 = 1$. Kako bi ovo važilo za proizvoljno x i y, neophodno je da važe naredni uslovi: $m_7 = m_8 = 0$ i $m_9 = 1$. Stoga ćemo nadalje razmatrati transformacije čije su matrice oblika:

$$\begin{bmatrix} m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 \\ 0 & 0 & 1 \end{bmatrix}$$

Ostaje pitanje na koji način na osnovu prethodno izvedenih matrica osnovnih geometrijskih transformacija dobiti matrice odgovarajućih transformacija u homogenim koordinatama. U slučaju linearnih transformacija dovoljno je „ugraditi“ postojeću matricu $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ u gornji levi ugao jedinične matrice dimenzije 3×3 i matrica linearne transformacije u homogenim koordinatama imaće oblik:

$$M = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Pokažimo istinitost ovog tvrđenja:

$$M \cdot v = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = v'$$

Što se translacije tiče, matrica translacije u homogenim koordinatama se može dobiti „ugrađivanjem“ vektora translacije u treću kolonu jedinične matrice:

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Naime, važi:

$$T \cdot v = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = v'$$

Primetimo da su koordinate tačke v pravilno translirane i da smo pritom, korišćenjem homogenih koordinata, translaciju izrazili u vidu množenja matrica.

Primer 3.2. Matrica skaliranja u odnosu na koordinatni početak za faktor 5 u smeru x ose i za faktor 7 u smeru y ose u terminima homogenih koordinata ima narednu formu:

$$S = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Matrica rotacije oko koordinatnog početka za $\pi/4$ glasi:

$$R = \begin{bmatrix} \cos(\pi/4) & -\sin(\pi/4) & 0 \\ \sin(\pi/4) & \cos(\pi/4) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Matrica translacije za -5 u smeru x ose i 7 u smeru y ose jednaka je:

$$T = \begin{bmatrix} 1 & 0 & -5 \\ 0 & 1 & 7 \\ 0 & 0 & 1 \end{bmatrix}$$

3.1.5 Inverzne transformacije

Ukoliko želimo da poništimo neku transformaciju, potrebno je da primenimo njoj inverznu transformaciju, odnosno da izvršimo množenje njoj inverznom matricom transformacije. Zahvaljujući homogenim koordinatama, sve pomenute matrice transformacija su invertibilne. Štaviše, inverzna transformacija svake od nabrojanih osnovnih transformacija je transformacija istog tipa.

Transformacija inverzna skaliranju u odnosu na koordinatni početak za faktor s_x po x osi i s_y po y osi sa matricom transformacije:

$$S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

je skaliranje za faktor $1/s_x$ po x osi i $1/s_y$ po y osi sa matricom transformacije:

$$S^{-1} = \begin{bmatrix} \frac{1}{s_x} & 0 & 0 \\ 0 & \frac{1}{s_y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Inverzna transformacija rotaciji oko koordinatnog početka za ugao φ čija je matrica transformacije jednaka:

$$R = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}$$

je rotacija oko koordinatnog početka za ugao $-\varphi$ kojoj odgovara matrica:

$$R^{-1} = \begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix}$$

Naime, inverz matrice rotacije je njen transponat, što se lako može proveriti njihovim množenjem.

Transformacija inverzna translaciji za vektor translacije $[t_x\ t_y]$ sa matricom transformacije:

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

je translacija za njemu suprotan vektor $[-t_x - t_y]$ kojoj odgovara matrica transformacije:

$$T^{-1} = \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Transformacija inverzna smicanju sa koeficijentom a u pravcu x koordinatne ose sa matricom transformacije:

$$SH_x = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

je smicanje sa koeficijentom $-a$ u pravcu x koordinatne ose kome odgovara matrica:

$$SH_x^{-1} = \begin{bmatrix} 1 & -a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Slično važi i za smicanje u pravcu y koordinatne ose.

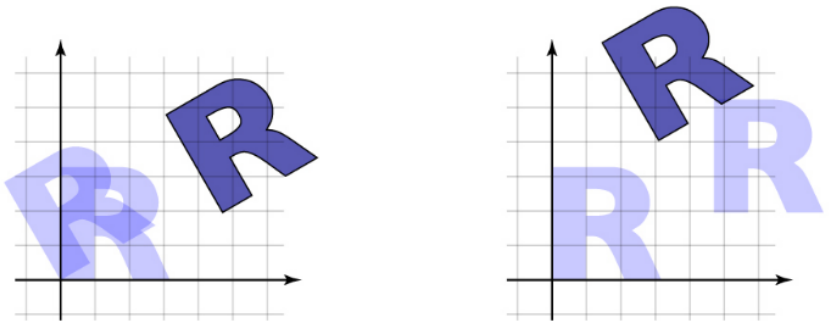
3.1.6 Kompozicija transformacija

Kada se na objekat na sceni primenjuje niz različitih geometrijskih transformacija, postavlja se pitanje na koji način predstaviti rezultujuću transformaciju. Transformacija je funkcija i važi da je $(f \cdot g)(i) = f(g(i))$. Ako funkcije f i g posmatramo kao matrice M_1 i M_2, a ulaz kao vektor v, kompozicija transformacija jednaka je $M_1 M_2 v$. Dakle, matrice složenih transformacija dobijamo tako što formiramo kompoziciju njihovih matrica transformacija.

Primer 3.3. Transformaciji kojom se tačka skalira u odnosu na koordinatni početak sa koeficijentima s_x i s_y, nakon toga rotira oko koordinatnog početka za ugao φ, a zatim translira za vektor $[t_x\ t_y]$, odgovarala bi naredna matrica transformacije:

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

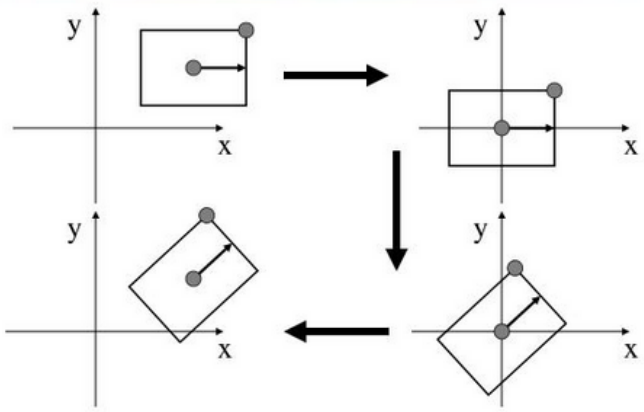
Podsetimo se da se matrice primenjuju zdesna ulevo i da množenje matrica u opštem slučaju nije komutativno, te u opštem slučaju transformacije ne komutiraju (slika 3.8). Na primer, ako izvršimo rotaciju za ugao φ oko koordinatnog početka, a zatim translaciju za vektor $[t_1\ t_2]$ dobijena transformacija razlikovaće se od transformacije kojom se najpre vrši translacija za vektor $[t_1\ t_2]$, a zatim rotacija za ugao φ. Međutim, u nekim specijalnim slučajevima transformacije komutiraju; na primer, translacije međusobno komutiraju; rotacije oko iste tačke međusobno komutiraju i sl.



Slika 3.8: Primer kada transformacije ne komutiraju: na slici levo prikazan je slučaj kada se slovo R čiji je donji-levi kraj u koordinatnom početku najpre rotira u odnosu na koordinatni početak pa translira, a na slici desno slučaj kada se prvo vrši translacija za isti vektor, a zatim rotacija u odnosu na koordinatni početak za taj isti ugao.

Primer 3.4. Neretko je potrebno izvršiti rotaciju oko tačke $P(x_1, y_1)$ (koja nije koordinatni početak) za ugao φ, na primer, izvršiti rotaciju oko koordinatnog početka koordinatnog sistema objekta. Ova transformacija se može realizovati na sledeći način:

- najpre se tačka P translira u koordinatni početak,
- zatim se vrši rotacija oko koordinatnog početka za ugao φ,
- potom se koordinatni početak translira u tačku P (slika 3.9).



Slika 3.9: Rotacija oko centra objekta (koji nije u koordinatnom početku).

Matrica odgovarajuće geometrijske transformacije jednaka je:

$$T_{x_1,y_1} \cdot R_\varphi \cdot T_{-x_1,-y_1} =$$
$$= \begin{bmatrix} 1 & 0 & x_1 \\ 0 & 1 & y_1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{bmatrix} =$$
$$= \begin{bmatrix} \cos\varphi & -\sin\varphi & x_1(1-\cos\varphi) + y_1\sin\varphi \\ \sin\varphi & \cos\varphi & y_1(1-\cos\varphi) - x_1\sin\varphi \\ 0 & 0 & 1 \end{bmatrix}$$

Ovim smo, u stvari, novi složeniji problem sveli na poznati problem – rotaciju oko koordinatnog početka. Na sličan način bi se izvela matrica skaliranja u odnosu na tačku koja nije koordinatni početak.

Primer 3.5. Razmotrimo problem u kome je dati objekat potrebno rotirati oko njegovog centra za $\pi/2$, a zatim ga skalirati 4 puta u svakom od smerova u odnosu na centar objekta. Problem ponovo rešavamo svođenjem na probleme koje znamo da rešimo. Naime, izvršićemo translaciju centra objekta u koordinatni početak, nakon toga rotaciju i skaliranje u odnosu na koordinatni početak, a zatim koordinatni početak translirati u centar objekta. Dakle, matrica transformacije jednaka je:

$$T^{-1} \cdot S_{4,4} \cdot R_{\pi/2} \cdot T$$

Ovu transformaciju treba primeniti na sva temena objekta koji je potrebno transformisati.

Kompozicija dve transformacije istog tipa je transformacija istog tipa. Specijalno, kompozicija dve translacije je translacija za zbir vektora translacija:

$$\begin{bmatrix} 1 & 0 & t'_x \\ 0 & 1 & t'_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & t''_x \\ 0 & 1 & t''_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t'_x + t''_x \\ 0 & 1 & t'_y + t''_y \\ 0 & 0 & 1 \end{bmatrix},$$

kompozicijom dva skaliranja u odnosu na koordinatni početak dobija se skaliranje u odnosu na koordinatni početak čiji su koeficijenti skaliranja jednaki proizvodu odgovarajućih koeficijenata skaliranja:

$$\begin{bmatrix} s'_x & 0 & 0 \\ 0 & s'_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s''_x & 0 & 0 \\ 0 & s''_y & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s'_x \cdot s''_x & 0 & 0 \\ 0 & s'_y \cdot s''_y & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

a kompozicijom dve rotacije oko koordinatnog početka dobija rotacija oko koordinatnog početka za ugao jednak zbiru uglova odgovarajućih rotacija:

$$\begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} \cos\alpha\cos\beta - \sin\alpha\sin\beta & -(\cos\alpha\sin\beta + \sin\alpha\cos\beta) & 0 \\ \sin\alpha\cos\beta + \cos\alpha\sin\beta & \cos\alpha\cos\beta - \sin\alpha\sin\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} \cos(\alpha+\beta) & -\sin(\alpha+\beta) & 0 \\ \sin(\alpha+\beta) & \cos(\alpha+\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Slično se pokazuje da se kompozicijom dva smicanja u pravcu iste koordinatne ose dobija smicanje u pravcu te koordinatne ose sa koeficijentom simicanja jednakim zbiru koeficijanata smicanja polaznih transformacija:

$$\begin{bmatrix} 1 & a_1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & a_2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & a_1+a_2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Inverz kompozicije transformacija je kompozicija inverznih matrica pojedinačnih transformacija u obrnutom redosledu:

$$(M_1 \cdot M_2 \cdot \ldots \cdot M_n)^{-1} = M_n^{-1} \cdot \ldots \cdot M_2^{-1} \cdot M_1^{-1}$$

Primer 3.6. Matrica inverzne transformacije prethodno razmatrane transformacije kojom se vrši rotacija i skaliranje u odnosu na centar objekta bila bi jednaka:

$$(T^{-1} \cdot S_{4,4} \cdot R_{\pi/2} \cdot T)^{-1} = T^{-1} \cdot R_{\pi/2}^{-1} \cdot S_{4,4}^{-1} \cdot (T^{-1})^{-1} = T^{-1} \cdot R_{-\pi/2} \cdot S_{1/4,1/4} \cdot T$$

3.1.7 Izometrijske i afine transformacije

Izometrijske transformacije čuvaju uglove i dužine i poznate su i pod nazivom *transformacije čvrstih tela* (eng. rigid-body transformation). Ovakve transformacije čuvaju oblik i veličinu objekata. Na primer, kvadrat stranice 1 sa stranicama paralelnim koordinatnim osama ostaje kvadrat (ne postaje ni romb niti pravougaonik) već može promeniti samo svoj položaj i ugao u odnosu na koordinatne ose. Rotacija i translacija spadaju u izometrijske transformacije, kao i proizvoljni niz rotacija i/ili translacija.

Za matricu kažemo da je *ortogonalna* ako vektori koji čine njene kolone predstavljaju ortonormiranu bazu tj. ako je intenzitet svake kolone jednak 1, a skalarni proizvod svake dve kolone jednak 0. Analogno svojstvo važi i za vrste ortogonalne matrice. Svakoj izometrijskoj transformaciji odgovara matrica transformacije oblika:

$$\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

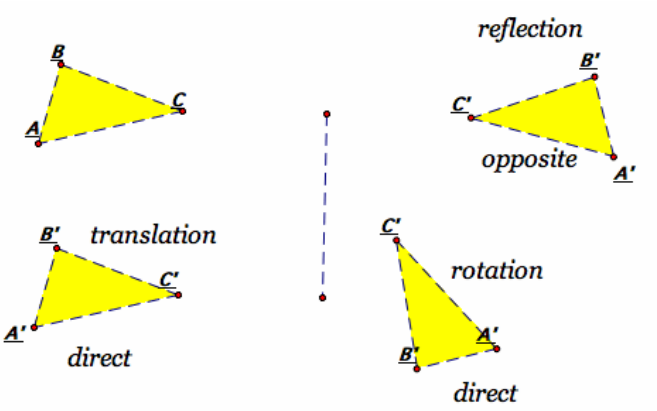
čija je gornja leva podmatrica dimenzije 2×2 ortogonalna, tj. važi:

- $a^2 + c^2 = 1$
- $b^2 + d^2 = 1$
- $ab + cd = 0$

Važi i obratno, matricama ovog oblika odgovaraju izometrijske transformacije.

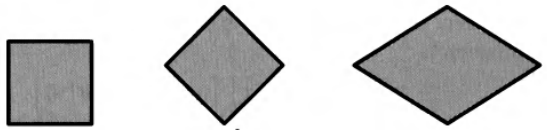
Izometrijske transformacije u ravni mogu se podeliti na *direktne* (eng. direct) i *indirektne* (eng. opposite). Direktne izometrijske transformacije čuvaju orijentaciju ravni, odnosno svaku trojku tačaka preslikavaju u trojku tačaka iste orijentacije. Primeri direktnih izometrijskih transformacija su translacija i rotacija. Ukoliko se prilikom izometrijske transformacije menja orijentacija tačaka, onda takvu izometrijsku transformaciju nazivamo indirektnom. Na primer, osnom refleksijom se menja orijentacija tačaka, te ona spada u indirektne izometrije (slika 3.10). Da bi se za izometrijsku transformaciju u ravni utvrdilo da li je direktna ili indirektna, dovoljno je izvršiti proveru orijentacije za jednu izabranu trojku tačaka.

Kompozicija dve indirektne izometrije je direktna izometrija, dok je kompozicija jedne direktne i jedne indirektne – indirektna izometrija.



Slika 3.10: Primeri direktnih i indirektnih izometrijskih transformacija.

Skup transformacija koje smo do sada razmatrali poznat je pod nazivom *afine transformacije* (eng. affine transformation). Afine transformacije su transformacije koje čuvaju kolinearnost tačaka (tj. slikaju prave u prave), odnose rastojanja između kolinearnih tačaka (npr. središte neke duži slika se u središte slike te duži) i paralelnost (paralelne prave ostaju paralelne). Afine transformacije ne čuvaju nužno uglove i dužine. Na slici 3.11 prikazano je dejstvo rotacije za ugao od $45°$ na kvadrat jedinične dimenzije, a zatim i efekat primene skaliranja sa koeficijentima 2 i 1. Primetimo da su paralelne prave ostale paralelne, ali se dužine duži i veličine uglova nisu očuvale. Dakle, afine transformacije ne čuvaju nužno oblik objekta.



Slika 3.11: Ilustracija dejstva afinih transformacija na jedinični kvadrat.

Translacija, rotacija, skaliranje i smicanje su afine transformacije i bilo koji niz translacija, rotacija, skaliranja i smicanja biće afina transformacija. Specijalno, sve izometrijske transformacije pripadaju afinim transformacijama.

S obzirom na to da se matrica afinih transformacija može zapisati kao:

$$\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} T^* & t^* \\ \mathbf{0} & 1 \end{bmatrix}$$

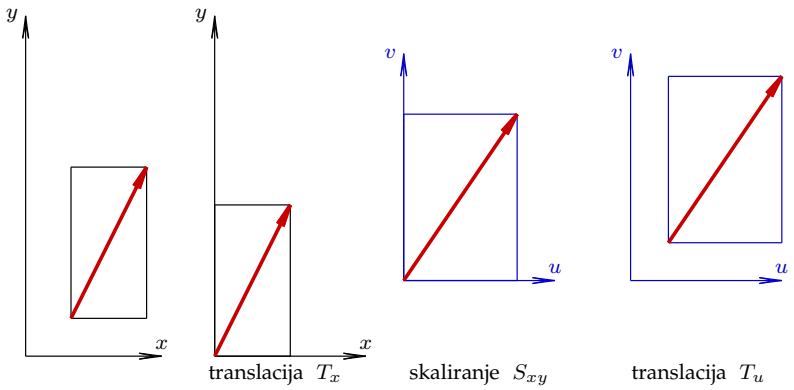
pri čemu je T^* matrica dimenzije 2×2, a t^* vektor kolona dimenzije 2, sledi da se svaka afina transformacija može predstaviti kao kompozicija jedne linearne transformacije (linearni deo afine transformacije) i jedne translacije:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = T^* \cdot \begin{bmatrix} x \\ y \end{bmatrix} + t^*$$

3.1.8 Preslikavanje slike u prozor

Preslikavanje figure iz jednog koordinatnog sistema u drugi je standardni problem u računarskoj grafici. Javlja se recimo kod preslikavanja slike iz nekog koordinatnog sistema u prozor ekrana.

Pretpostavimo da treba preslikati sadržaj pravougaonika sa stranicama paralelnim koordinatnim osama i levim donjim temenom (x_{min}, y_{min}) i gornjim desnim temenom (x_{max}, y_{max}) u koordinatnom sistemu (x, y) u pravougaonik sa stranicama paralelnim koordinatnim osama i sa levim donjim temenom (u_{min}, v_{min}) i gornjim desnim temenom (u_{max}, v_{max}) u koordinatnom sistemu (u, v) (slika 3.12). Ovo preslikavanje možemo realizovati kao kompoziciju tri osnovne transformacije. Naime, svaku tačku pravougaonika treba:



Slika 3.12: Preslikavanje iz jednog koordinatnog sistema u drugi.

- preslikati translacijom koja tačku za koordinatama (x_{min}, y_{min}) preslikava u koordinatni početak, čija je matrica jednaka:

$$T_x = \begin{bmatrix} 1 & 0 & -x_{min} \\ 0 & 1 & -y_{min} \\ 0 & 0 & 1 \end{bmatrix}$$

- preslikati skaliranjem u odnosu na koordinatni početak kojim se veličina prvog pravougaonika svodi na veličinu drugog pravougaonika: dužine stranica prvog pravougaonika su $(x_{max} - x_{min}, y_{max} - y_{min})$, a drugog $(u_{max} - u_{min}, v_{max} - v_{min})$, pa su faktori skaliranja redom jednaki $\frac{u_{max}-u_{min}}{x_{max}-x_{min}}$ i $\frac{v_{max}-v_{min}}{y_{max}-y_{min}}$, a odgovarajuća matrica jednaka je:

$$S_{xu} = \begin{bmatrix} \frac{u_{max}-u_{min}}{x_{max}-x_{min}} & 0 & 0 \\ 0 & \frac{v_{max}-v_{min}}{y_{max}-y_{min}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- preslikati translacijom koja koordinatni početak preslikava u tačku sa koordinatama (u_{min}, v_{min}), a čija je matrica jednaka:

$$T_u = \begin{bmatrix} 1 & 0 & u_{min} \\ 0 & 1 & v_{min} \\ 0 & 0 & 1 \end{bmatrix}$$

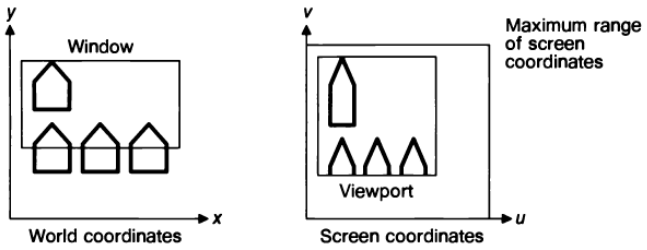
Kompozicija navedenih transformacija opisana je matricom koja je dobijena množenjem odgovarajućih matrica:

$$M_{xu} = T_u \cdot S_{xu} \cdot T_x = \begin{bmatrix} \frac{u_{max}-u_{min}}{x_{max}-x_{min}} & 0 & -x_{min}\frac{u_{max}-u_{min}}{x_{max}-x_{min}} + u_{min} \\ 0 & \frac{v_{max}-v_{min}}{y_{max}-y_{min}} & -y_{min}\frac{v_{max}-v_{min}}{y_{max}-y_{min}} + v_{min} \\ 0 & 0 & 1 \end{bmatrix}$$

Nekada se transformacija slike u prozor kombinuje sa operacijom odsecanja primitiva u odnosu na dati region (slika 3.13).

3.2 3D transformacije

Homogene koordinate za trodimenzioni prostor definišu se analogno homogenim koordinatama u ravni: tačka (x, y, z) trodimenzionog prostora predstavlja se četvorkom vrednosti (x, y, z, W). Dve četvorke vrednosti (x, y, z, W) i (x', y', z', W') predstavljaju istu tačku ako i samo ako je jedna četvorka umnožak druge, odnosno ako postoji vrednost t tako da je $x' = tx$, $y' = ty$, $z' = tz$ i $W' = tW$. Tako četvorke vrednosti $(1, 2, 3, 4)$ i $(2, 4, 6, 8)$ predstavljaju jednu istu tačku. Pritom bar jedna od vrednosti x, y, z i W mora biti različita od 0, tj. četvorka vrednosti $(0, 0, 0, 0)$ nije dozvoljena. Standardna reprezentacija tačke sa homogenim koordinatama (x, y, z, W) za $W \neq 0$ je $(x/W, y/W, z/W, 1)$, i prve tri koordinate predstavljaju Dekartovske koordinate tačke.



Slika 3.13: Primitive u svetskom koordinatnom sistemu se odsecaju u odnosu na prozor i samo one koje pripadaju prozoru se prikazuju na ekranu.

Homogene koordinate nam omogućavaju da na uniforman način predstavimo i tačke i vektore u trodimenzionom prostoru – kao četvorke vrednosti (x, y, z, W). Pritom, postoji vrlo jasna razlika između njih, naime, kod tačaka je homogena koordinata W različita od 0, dok je kod vektora ona jednaka 0. Često ćemo mi na vektore u trodimenzionom prostoru referisati i kao na tačke u beskonačnosti.

Kada se koriste homogene koordinate, transformacije u trodimenzionom prostoru se predstavljaju matricama dimenzije 4×4. Njih je moguće primeniti i na tačke i na vektore.

3.2.1 Osnovne transformacije

Razmotrimo matrice osnovnih transformacija u trodimenzionom prostoru.

Translacija

Matricu translacije u trodimenzionom prostoru izraženu u homogenim koordinatama dobijamo kao proširenje matrice translacije u ravni – vektor translacije $t = [t_x\ t_y\ t_z]^T$ je u ovom slučaju dužine 3 i potrebno ga je „ugraditi“ u poslednju kolonu jedinične matrice dimenzije 4×4:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Odgovarajuće jednačine translacije glase:

$$\begin{aligned} x' &= x + t_x \\ y' &= y + t_y \\ z' &= z + t_z \end{aligned}$$

Kao i u slučaju translacije u ravni, važi svojstvo da je kompozicija dve translacije translacija i da je transformacija inverzna translaciji za vektor t translacija za vektor $-t$.

Skaliranje

Razmotrimo kakav će format imati matrica skaliranja u odnosu na koordinatni početak u slučaju trodimenzionog prostora. Matrica je, kao i u slučaju odgovarajuće transformacije u ravni, dijagonalna i na glavnoj dijagonali nalaze se redom faktori skaliranja u odnosu na x, y i z osu:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

odnosno jednačine skaliranja u trodimenzionom prostoru glase:

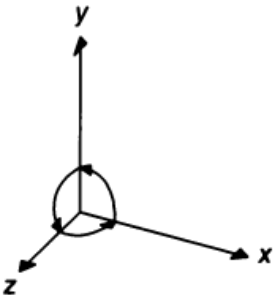
$$\begin{aligned} x' &= s_x \cdot x \\ y' &= s_y \cdot y \\ z' &= s_z \cdot z \end{aligned}$$

Kompozicija dva skaliranja u odnosu na koordinatni početak je skaliranje u odnosu na koordinatni početak i transformacija inverzna skaliranju u odnosu na koordinatni početak je takođe skaliranje u odnosu na koordinatni početak.

Matrica skaliranja u odnosu na proizvoljnu tačku P (različitu od koordinatnog početka) može se dobiti analogno skaliranju u ravni: translacijom tačke P u koordinatni početak, skaliranjem u odnosu na koordinatni početak sa istim koeficijentima skaliranja, a potom translacijom koordinatnog početka u tačku P.

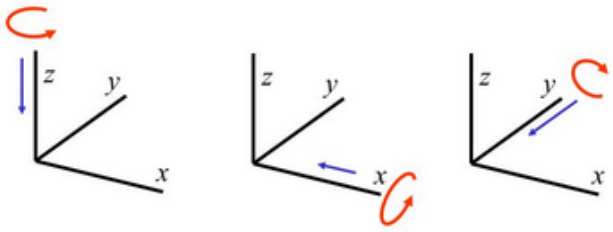
Rotacija

Najveća razlika u transformacijama u ravni i prostoru nastaje u slučaju rotacije. Dok smo u ravni razmatrali rotaciju oko proizvoljne tačke ali je osa rotacije bila fiksna (upravna na ravan), u trodimenzionom prostoru možemo razmatrati rotaciju oko proizvoljnog vektora, odnosno ose $\vec{u} = [u_x \;\; u_y \;\; u_z]^T$.



Slika 3.14: Pozitivno orijentisan koordinatni sistem.

Pretpostavimo da je koordinatni sistem pozitivno orijentisan, odnosno da za njega važi pravilo desne ruke. Za pozitivno orijentisani koordinatni sistem važi: ako je osa rotacije x osa, onda je pozitivan smer rotacije od pozitivnog dela y ose ka pozitivnom delu z ose; slično, ako je osa rotacije y osa, onda je pozitivan smer rotacije od pozitivnog dela z ose ka pozitivnom delu x ose i analogno, ako je osa rotacije z osa, onda je pozitivan smer rotacije od pozitivnog dela x ose ka pozitivnom delu y ose (slika 3.14). Dakle, uvek ćemo razmatrati isti ciklični poredak koordinata: x, y, z (odnosno y, z, x i z, x, y).



Slika 3.15: Ilustracija rotacija oko koordinatnih osa.

Razmotrimo najpre rotacije oko koordinatnih osa (slika 3.15) i njihove matrice transformacija. Matricu rotacije oko z koordinatne ose možemo izvesti razmatranjem na koji način ona deluje na vektore standardne baze: naime, njom se jedinični vektor u smeru z ose ne menja, dok jedinični vektori u smeru x i y ose ostaju u ravni Oxy (z koordinata će im biti nula) dok se x i y koordinate menjaju na isti način kao u slučaju ravni. Dakle, rotacija oko z ose se u homogenim koordinatama može opisati narednom matričnom jednačinom:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 & 0 \\ \sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

dok odgovarajuće jednačine rotacije oko z koordinatne glase glase:

$$\begin{aligned} x' &= x \cdot \cos\varphi - y \cdot \sin\varphi \\ y' &= x \cdot \sin\varphi + y \cdot \cos\varphi \\ z' &= z \end{aligned}$$

Dakle, z koordinate tačaka se ne menjaju, a x i y koordinate se menjaju na isti način kao u slučaju rotacije u ravni. Kompozicija dve rotacije oko z ose je ponovo rotacija oko z ose.

Analogno se izvode i matrica rotacije oko x ose:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi & 0 \\ 0 & \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

i matrica rotacije oko y ose:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\varphi & 0 & \sin\varphi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\varphi & 0 & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

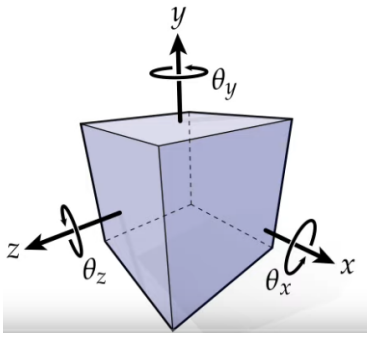
Istaknimo i to da rotacije u trodimenzionom prostoru u opštem slučaju međusobno ne komutiraju: ako objekat recimo rotiramo oko x ose, pa oko y ose, nećemo dobiti isto kao ako najpre izvršimo rotaciju oko y ose, pa oko x ose.

Na koji način doći do matrice rotacije oko proizvoljne ose za dati ugao? To možemo uraditi indirektno, svođenjem na jednostavniji problem ili direktno.

Razmotrimo ponovo rotaciju oko proizvoljne ose u za dati ugao Ψ. Kao i u dvodimenzionom slučaju, problem rešavamo tako što ga svodimo na problem koji umemo da rešimo. Osu u dovodimo na poklapanje sa jednom od koordinatnih osa (na primer, x osom), zatim vršimo rotaciju oko te koordinatne ose za ugao Ψ, a nakon toga vršimo preslikavanje te koordinatne ose na osu u. Dakle, postupak se može opisati narednim nizom koraka:

1. odrediti ugao Θ za koji treba rotirati osu u oko y ose kako bismo je oborili u ravan Oxy,
2. odrediti ugao Φ za koji treba rotirati dobijenu sliku ose u oko z ose kako bi je doveli na poklapanje sa x osom,
3. sad kada je vektor u poravnat sa x osom treba izvršiti rotaciju objekta za ugao Ψ oko x ose,
4. izvršiti inverzne transformacije transformacijama poravnanja (rotacijama oko z i y koordinatne ose).

Razmotrimo sada na koji način možemo direktno zadati rotaciju u 3D prostoru. Svaka rotacija se može predstaviti kao kompozicija tri rotacije oko tri različite ose: oko x ose u Oyz ravni za ugao θ_x, oko y ose u Oxz ravni za ugao θ_y i oko z ose u Oxy ravni za ugao θ_z (eng. pitch, yaw, roll). Ovi uglovi nazivaju se *Ojlerovi uglovi* (slika 3.16) i često se koriste u simulatorima letenja.



Slika 3.16: Ilustracija Ojlerovih uglova.

Razmotrimo kompoziciju rotacije oko z, y i x ose. Njena matrica jednaka je:

$$\begin{aligned} R &= R_x R_y R_z \\ &= \begin{bmatrix} \cos\theta_y\cos\theta_z & -\cos\theta_y\sin\theta_z & \sin\theta_y \\ \cos\theta_z\sin\theta_x\sin\theta_y + \cos\theta_x\sin\theta_z & \cos\theta_x\cos\theta_z - \sin\theta_x\sin\theta_y\sin\theta_z & -\cos\theta_y\sin\theta_x \\ -\cos\theta_x\cos\theta_z\sin\theta_y + \sin\theta_x\sin\theta_z & \cos\theta_z\sin\theta_x + \cos\theta_x\sin\theta_y\sin\theta_z & \cos\theta_x\cos\theta_y \end{bmatrix} \end{aligned} \quad (3.1)$$

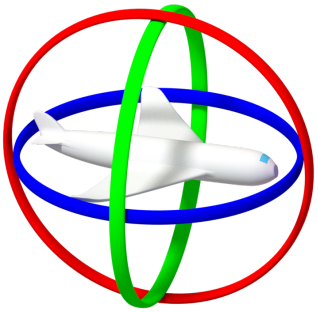
Primetimo da tačan oblik matrice rotacije R zavisi od redosleda matrica u ovoj kompoziciji. Na osnovu matrice rotacije (3.1) nije teško odrediti uglove θ_x, θ_y i θ_z:

$$\begin{aligned} \theta_y &= \arcsin(R_{02}) \\ \theta_z &= -\arctan\frac{R_{01}}{R_{00}} \\ \theta_x &= -\arctan\frac{R_{12}}{R_{22}} \end{aligned}$$

Popularnost Ojlerovih uglova leži u njihovoj jednostavnosti i intuitivnoj reprezentaciji. Međutim, ova vrsta reprezentacije 3D rotacije pati od jednog velikog problema. Razmotrimo situaciju kada ugao θ_y postane jednak $\pi/2$, važiće $\cos\theta_y = 0$ i $\sin\theta_y = 1$ i matrica R se pojednostavljuje:

$$\begin{aligned} R &= \begin{bmatrix} 0 & 0 & 1 \\ \cos\theta_z\sin\theta_x + \cos\theta_x\sin\theta_z & \cos\theta_x\cos\theta_z - \sin\theta_x\sin\theta_z & 0 \\ -\cos\theta_x\cos\theta_z + \sin\theta_x\sin\theta_z & \cos\theta_z\sin\theta_x + \cos\theta_x\sin\theta_z & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 1 \\ \sin(\theta_x+\theta_z) & \cos(\theta_x+\theta_z) & 0 \\ -\cos(\theta_x+\theta_z) & \sin(\theta_x+\theta_z) & 0 \end{bmatrix} \end{aligned}$$

Na ovaj način dobijamo matricu rotacije u jednoj ravni, odnosno nezavisno menjanje vrednosti ugla θ_x i θ_z utiče na isti način na rotaciju. U opštem slučaju problem koji se može javiti pri nezavisnom podešavanju rotacije oko svake od osa jeste taj što možemo doći u situaciju kada je nemoguće vršiti rotaciju oko neke od tri koordinatnih osa. Ovaj problem se naziva *gimbal lock* i posledica je nepogodne matematičke parametrizacije rotacije. Na ovaj način gubi se jedan stepen slobode u mehanizmu sa tri osovine (slika 3.17). Ovo nije pogodan dizajn za npr. kontrole aviona.



Slika 3.17: Mehanizam postizanja proizvoljne rotacije pojedinačnim podešavanjem vrednosti rotacije oko svake od koordinatnih osa.

U opštem slučaju, matrica rotacije oko proizvoljne ose u za ugao φ data je narednom formulom:

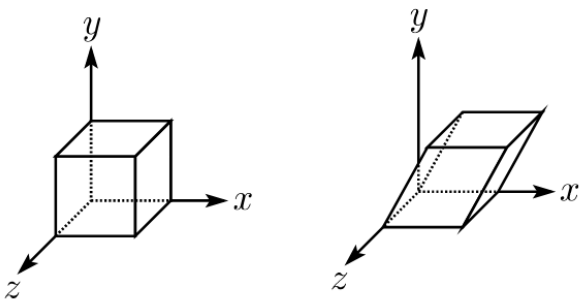
$$\begin{bmatrix} \cos\varphi + u_x^2(1-\cos\varphi) & u_xu_y(1-\cos\varphi) - u_z\sin\varphi & u_xu_z(1-\cos\varphi) + u_y\sin\varphi \\ u_xu_y(1-\cos\varphi) + u_z\sin\varphi & \cos\varphi + u_y^2(1-\cos\varphi) & u_yu_z(1-\cos\varphi) - u_x\sin\varphi \\ u_xu_z(1-\cos\varphi) - u_y\sin\varphi & u_yu_z(1-\cos\varphi) + u_x\sin\varphi & \cos\varphi + u_z^2(1-\cos\varphi) \end{bmatrix}$$

Ova formula poznata je kao *Rodrigezova formula* i nećemo je izvoditi.
Do rotacije u 3D prostoru možemo doći i posredstvom *kvaterniona* (eng. quaternion).

Smicanje

U trodimenzionom prostoru smicanje podrazumeva iskošenje objekta u pravcu paralelnom nekoj koordinatnoj ravni. Kao i u slučaju smicanja u ravni, vrednosti jedne koordinate ostaju fiksne. Na slici 3.18 prikazano je dejstvo transformacije smicanja u pravcu Oxz koordinatne ravni na jediničnu kocku: strana kocke koja leži u Oxz ravni ostaje neizmenjena, dok se strana koja pripada ravni $y = 1$ translira za odgovarajući vektor.

Razmotrimo na koji način smicanje u pravcu koordinatne ravni Oxz deluje na vektore standardne baze: jedinični vektori u smeru x i z koordinatne ose slikaju se u njih same, a jedinični vektor u smeru



Slika 3.18: Transformacija smicanja u trodimenzionom prostoru.

y koordinatne ose slika se u vektor čija je y koordinata jednaka 1, dok vrednosti koordinata x i z zavise od koeficijenata smicanja. Dakle, matrica smicanja u pravcu koordinatne ravni Oxz u homogenim koordinatama ima formu:

$$\begin{bmatrix} 1 & a & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & b & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

odnosno, odgovarajuće jednačine smicanja glase:

$$\begin{aligned} x' &= x + a \cdot y \\ y' &= y \\ z' &= z + b \cdot y \end{aligned}$$

Nekada se kaže i da se smicanje vrši duž neke koordinatne ose, u ovom slučaju duž y ose. Transformacija inverzna smicanju je opet smicanje duž iste koordinatne ravni.

Transformacija smicanja je korisna za generisanje projekcija, na primer, može se iskoristiti za crtanje trodimenzione kocke na dvodimenzionom ekranu.

3.2.2 Kompozicije afinih transformacija

Svakoj kompoziciji rotacija, skaliranja, translacija i smicanja odgovara matrica oblika:

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R^* & t^* \\ \mathbf{0} & 1 \end{bmatrix}$$

gde je matrica R^* dimenzije 3×3 i opisuje kombinovani efekat rotacija, skaliranja i smicanja, a t^* vektor dimenzije 3 koji opisuje kombinovani efekat svih translacija.

U skladu sa ovim razmatranjem, umesto množenja vektora homogenih koordinata tačke matricom dimenzije 4×4, može se koristiti efikasnije izračunavanje:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R^* \begin{bmatrix} x \\ y \\ z \end{bmatrix} + t^*$$

Matrice transformacija određuju slike pojedinačnih tačaka. S obzirom na to da afine transformacije slikaju prave u prave, a ravni u ravni, slika prave pri nekoj afinoj transformaciji se može odrediti kao prava određena slikama dve tačke polazne prave, a slika ravni kao ravan određena slikama tri nekolinearne tačke polazne ravni (ili kao ravan koja sadrži sliku neke tačke i ima dati vektor normale).

3.3 Transformacije kao promena koordinatnog sistema

Do sada su transformacije korišćene za preslikavanje skupa tačaka jednog koordinatnog sistema u taj isti koordinatni sistem. U tom kontekstu, možemo reći da je objekat transformisan, a da je koordinatni

sistem ostao isti. S druge strane, istu transformaciju možemo opisati i kao promenu koordinatnog sistema, pri čemu smatramo da se objekat ne transformiše, već se samo računaju njegove koordinate u novom koordinatnom sistemu. Prvi stil razmišljanja može biti pogodniji kada se objekat kreće, a drugi, recimo, kada više objekata u svojim pojedinačnim koordinatnim sistemima treba objediniti i izraziti njihove koordinate u jedinstvenom svetskom koordinatnom sistemu.

Za promenu koordinatnog sistema mogu se koristiti iste tehnike kao i za transformacije objekta u okviru jednog istog koordinatnog sistema. Označimo sa $M_{i\leftarrow j}$ matricu transformacije koja prevodi koordinatni sistem j u koordinatni sistem i. Označimo sa $P^{(i)}$ koordinate tačke P u koordinatnom sistemu i, sa $P^{(j)}$ koordinate tačke P u koordinatnom sistemu j, a sa $P^{(k)}$ koordinate u koordinatnom sistemu k. Tada važi:

$$P^{(i)} = M_{i\leftarrow j}P^{(j)} = M_{i\leftarrow j}M_{j\leftarrow k}P^{(k)} = M_{i\leftarrow k}P^{(k)}$$

odakle sledi

$$M_{i\leftarrow j}M_{j\leftarrow k} = M_{i\leftarrow k} \tag{3.2}$$

Ako bismo i levu i desnu stranu jednačine

$$P^{(i)} = M_{i\leftarrow j}P^{(j)}$$

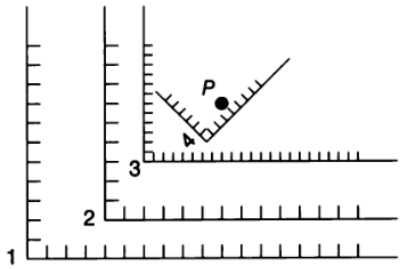
pomnožili sa leve strane matricom $M^{-1}_{i\leftarrow j}$ dobili bismo:

$$M^{-1}_{i\leftarrow j}P^{(i)} = P^{(j)}$$

odakle sledi da je matrica $M^{-1}_{i\leftarrow j}$ matrica transformacije iz koordinatnog sistema i u koordinatni sistem j, odnosno da važi:

$$M_{j\leftarrow i} = M^{-1}_{i\leftarrow j} \tag{3.3}$$

Na osnovu jednakosti (3.2) i (3.3) možemo zaključiti da je za proizvoljnu promenu koordinatnog sistema dovoljno imati opis svođenja iz različitih koordinatnih sistema na jedan (proizvoljni) koordinatni sistem.



Slika 3.19: Četiri različita koordinatna sistema u ravni.

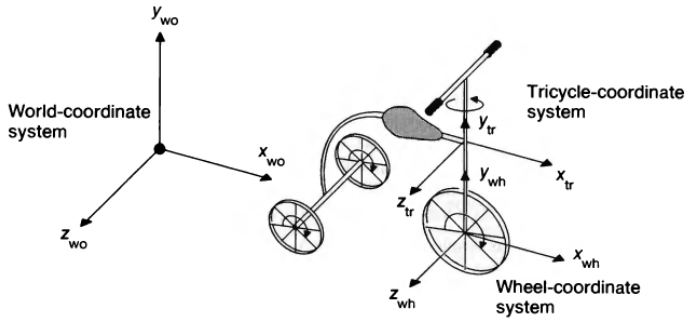
Na slici 3.19 prikazana su četiri različita koordinatna sistema u ravni.

Razmotrimo kako bi izgledale matrice transformacija iz jednog u drugi koordinatni sistem. Možemo jednostavno zaključiti da je matrica transformacije iz koordinatnog sistema 2 u koordinatni sistem 1 jednaka $M_{1\leftarrow 2} = T_{4,2}$ i slično $M_{2\leftarrow 3} = T_{2,3} \cdot S_{0.5,0.5}$ i $M_{3\leftarrow 4} = T_{6.7,1.8} \cdot R_{\pi/4}$, te stoga važi $M_{1\leftarrow 3} = M_{1\leftarrow 2} \cdot M_{2\leftarrow 3} = T_{4,2} \cdot T_{2,3} \cdot S_{0.5,0.5}$.

S obzirom na to da važi $M_{1\leftarrow 2} = T_{4,2}$, važi i $M_{2\leftarrow 1} = T_{-4,-2}$, pa pošto tačka P sa slike 3.19 u prvom koordinatnom sistemu ima koordinate $P^{(1)} = (10, 8)$, važi i $P^{(2)} = T_{-4,-2} \cdot P^{(1)} = (6, 6)$. Slično se dobija da je matrica transformacije iz koordinatnog sistema 2 u koordinatni sistem 3 jednaka $M_{3\leftarrow 2} = (T_{2,3} \cdot S_{0.5,0.5})^{-1} = S_{2,2} \cdot T_{-2,-3}$, pa iz toga što važi $P^{(2)} = (6, 6)$ sledi $P^{(3)} = S_{2,2} \cdot T_{-2,-3} \cdot P^{(2)} = (8, 6)$.

Primer 3.7. Interesantan primer transformacije koordinatnog sistema jeste ona koja transformiše pozitivno orijentisan koordinatni sistem u negativno orijentisan koordinatni sistem i njena matrica ima oblik:

$$M_{R\to L} = M_{L\to R} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Slika 3.20: Tricikl zadat u dva različita koordinatna sistema.

Primer 3.8. Tricikl je opisan u svetskom koordinatnom sistemu i želimo da odredimo koordinate jedne tačke na prednjem točku dok se tricikl pravolinijski kreće u smeru x koordinatne ose (slika 3.20). Na tačku sa točka tricikla delovaće rotacija oko njegove ose zbog okretanja točka, kao i translacija, s obzirom da se tricikl kreće. Odaberimo najpre neki pogodniji koordinatni sistem za opisivanje ovog kretanja, na primer, onaj u kojem centar točka predstavlja koordinatni početak, z koordinatna osa sadrži osu točka, dok se x koordinatna osa poklapa sa smerom kretanja tricikla. U ovom koordinatnom sistemu nije teško odrediti koordinate tražene tačke prilikom kretanja tricikla. Naime, ako se točak zarotira za ugao α, tačka na točku preći će put jednak αR, gde je sa R označen poluprečnik točka. Dakle nove koordinate tačke P u polaznom koordinatnom sistemu točka biće jednake:

$$P' = T_{\alpha R,0,0} \cdot R_z(\alpha) \cdot P$$

dok će njene koordinate u novom (transliranom) koordinatnom sistemu točka biti jednake:

$$P' = R_z(\alpha) \cdot P$$

Koordinate u polaznom koordinatnom sistemu možemo zatim izračunati primenom matrice transformacije jednog koordinatnog sistema u drugi.

3.4 Graf scene

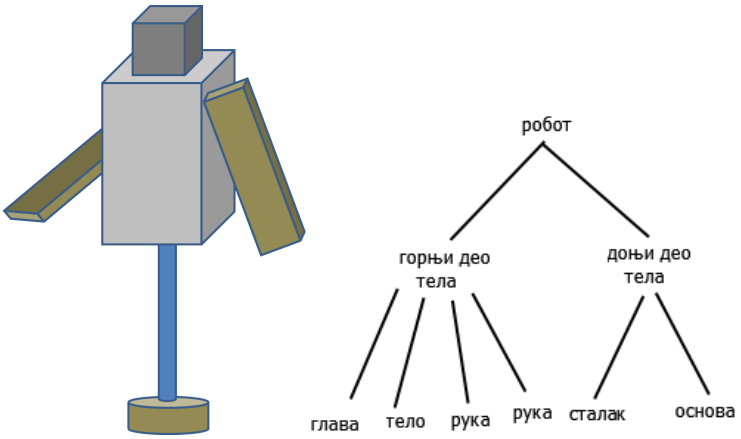
Scena u računarskoj grafici može biti izuzetno složena i sadržati veliki broj objekata od kojih svaki može biti složene strukture i imati svoja svojstva, odnosno atribute. Stoga je veoma važno osmisliti efikasnu strukturu podataka za predstavljanje sadržaja scene, njenu izmenu tokom vremena, kao i renderovanje sadržaja scene.

Kao što je već pomenuto, geometrijske transformacije se između ostalog koriste i za postavljanje veličine objekta i za pozicioniranje objekata na sceni. Za svaki objekat na sceni moguće je zadati njegovu transformaciju pojedinačnom matricom: na taj način scenu bismo modelovali kao stablo dubine 1 u kome koren ima onoliko dece koliki je broj objekata na sceni. Renderovanje scene zadate na ovaj način biće dosta brzo, međutim, ako se više objekata na sceni kreće, potrebno je izmeniti transformacije za sve ove objekte.

Primetimo da su objekti najčešće hijerarhijske strukture i da su sačinjeni od komponenti. Na primer, ako posmatramo robota sa slike 3.21, on se sastoji od gornjeg i donjeg dela, a svaki od ova dva dela od pojedinačnih komponenti. Dakle, objekat je pogodno definisati kao hijerarhijsku strukturu te na taj način omogućiti transformaciju većeg broja komponenti kao celine primenom samo jedne transformacije na čitavu grupu komponenti (na primer, pozicioniranjem kompletnog gornjeg dela robota sačinjenog od većeg broja elemenata).

Graf scene (eng. scene graph) je hijerarhijska struktura podataka pogodna za predstavljanje logičke i prostorne reprezentacije trodimenzione scene. Reprezentacija grafa scene treba da bude pogodna za zadavanje velikog broja objekata u složenim i dinamičnim scenama. Najjednostavnija struktura grafa scene je u vidu stabla.

Sama struktura grafa scene nije strogo definisana, odnosno nije precizno zadato koje elemente graf scene može a koje ne može sadržati, već se njegova struktura može prilagoditi pojedinačnoj situaciji. Prilikom dizajniranja grafa scene potrebno je razmotriti nekoliko važnih pitanja. Na primer, da li su transformacije deo čvora objekta ili predstavljaju zasebne čvorove u grafu scene ili se pak nalaze na



Slika 3.21: Model robota i jedna njegova moguća reprezentacija u vidu grafa scene.

granama grafa scene. Slično pitanje se može postaviti u vezi atributa. Takođe, postavlja se pitanje da li se vrednost atributa primenjuje samo na čvor čije je to svojstvo ili ga nasleđuju sva deca tog čvora.

Graf scene se uobičajeno sastoji iz dva tipa čvorova:

- listova – koji predstavljaju objekte koji imaju ugrađenu geometriju i koji se mogu renderovati: to su, recimo, kocke, valjci, lopte; ovi objekti su podrazumevano jedinične veličine i pozicionirani su u koordinatnom početku,
- unutrašnjih čvorova – koji mogu imati veći broj dece i koji se koriste za izmenu stanja poput primene transformacije ili nekog svojstva.

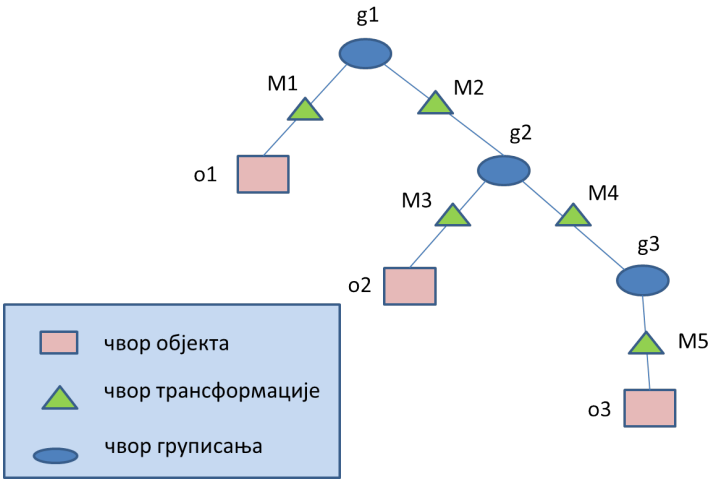
U korenu grafa scene nalazi se svet koji modelujemo, odnosno čitav sadržaj trodimenzione scene. Mi ćemo razmatrati slučaj kada u grafu scene postoje čvorovi transformacija. Transformacije se mogu primeniti na pojedinačne objekte (koji odgovaraju listovima grafa scene) ili na grupe objekata (koji su predstavljeni unutrašnjim čvorovima grafa scene). Pored čvorova objekata i čvorova transformacija u graf scene uvodimo i čvorove grupisanja koji su unutrašnji čvorovi grafa. Ovi čvorovi predstavljaju logičku grupu objekata. U njima se ne čuvaju nikakvi geometrijski podaci, već se njima omogućavaju izmene na višem nivou hijerarhije primenom transformacije na čvor odgovarajuće grupe, a takođe omogućavaju i ponovno korišćenje grupe kao celine na nekom drugom mestu u grafu scene. Svaki list grafa scene predstavlja objekat ili deo objekata i sadrži sve neophodne geometrijske informacije. Kamera i izvori svetla se takođe mogu predstaviti listovima grafa scene.

Ovakva struktura grafa scene omogućava da umesto da za svaki novi objekat koji nam je potreban pravimo novu primitivu, možemo koristiti minimalan broj osnovnih primitiva, a složenije oblike kreirati primenama transformacija na ovaj manji skup primitiva. Na primer, elipsu dobijamo primenom odgovarajuće transformacije skaliranja na krug jediničnog poluprečnika. Pritom je svaka od primitiva zadata u svom koordinatnom sistemu i pogodnosti radi u svojoj kanonskoj poziciji i orijentaciji: na primer, kocku bismo zadali tako da joj jedno teme bude u koordinatnom početku a da joj strane naležu na koordinatne ravni.

Kao što smo već pomenuli, svaka transformacija deluje na svu geometriju definisanu ispod nje. Kumulativna matrica transformacije koja deluje na neku primitivu gradi se penjanjem uz stablo počev od te primitive do korena i množenjem matrica transformacija koje se javljaju duž tog puta, tako što se transformacije na višem nivou dodaju na početak niza.

Primer 3.9. Ako razmotrimo graf scene sa slike 3.22 kumulativna matrica transformacije za objekat o_1 jednaka je M_1, za objekat o_2 jednaka je $M_2 \cdot M_3$, a za objekat o_3 iznosi $M_2 \cdot M_4 \cdot M_5$.

Kao što smo već napomenuli, najjednostavnija struktura grafa scene je predstavljena stablom. Međutim, jedna od osnovnih karakteristika hijerarhijskog modelovanja je *instanciranje*, odnosno ponovno korišćenje komponenti. Bilo bi dobro omogućiti ponovno korišćenje već definisanih grupa objekata. Ovo je posebno korisno ako imamo više sličnih komponenti u sceni kao, na primer, dve ruke robota ili četiri točka automobila. Ovo odgovara situaciji u kojoj jedan čvor može da ima veći broj roditelja, odnosno scenariju kada jedan objekat može biti član većeg broja grupa. Na ovaj način graf scene prestaje da bude stablo. Međutim, pošto je odnos roditelj-dete jednosmeran i dete ne može biti predak samom sebi, graf

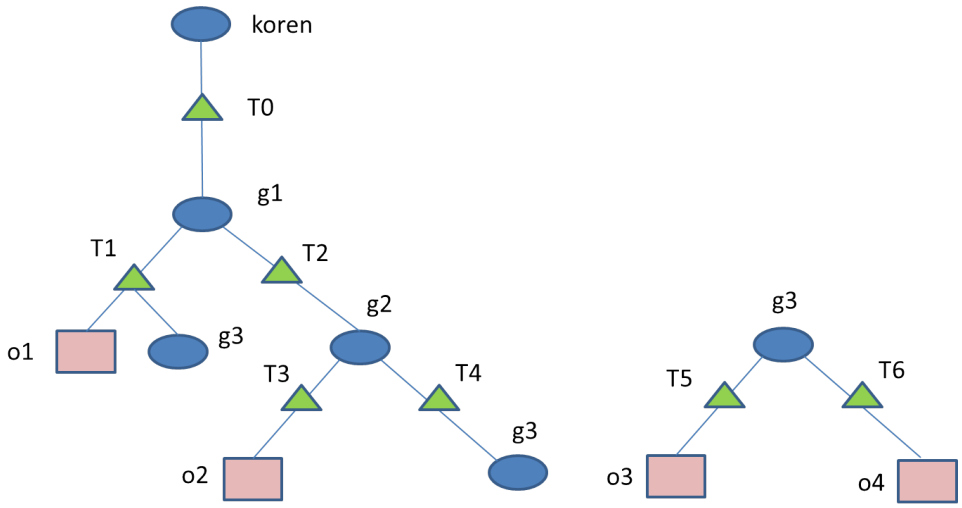


Slika 3.22: Primer grafa scene u obliku stabla.

scene u ovom slučaju ima oblik usmerenog acikličkog grafa (eng. DAG, directed acyclic graph). Dakle, u ovom slučaju graf scene nije stablo, ali mu nalikuje, jedina razlika je ta što se neke grane spajaju, čime je moguće na različite načine doći od korena do jednog istog lista grafa.

Na slici 3.23 prikazan je graf scene koji nema oblik stabla jer se grupa g_3 koristi na dva mesta u grafu. Ova situacija odgovara tome da čvor g_3 ima dva roditeljska čvora: čvorove T_1 i T_4. Transformacije definisane u okviru grupe g_3 se ne menjaju, ali se kumulativne matrice transformacije za svako korišćenje grupe g_3 kao celine razlikuju.

Primer 3.10. Kumulativna matrica transformacije za prvo korišćenje grupe g_3 jednaka je $T_0 \cdot T_1$, dok je za drugo korišćenje ove grupe ona jednaka $T_0 \cdot T_2 \cdot T_4$.



Slika 3.23: Primer grafa scene koji nije u obliku stabla.

Objekat pored svoje geometrije ima i druga svojstva poput boje, parametara senčenja, transparentnosti ili detalja teksture, te se graf scene može obogatiti i čvorovima atributa. Graf scene može sadržati i čvorove kojima se zadaju izvori svetla jer je i njih potrebno pozicionirati na sceni i zadati im usmerenje, a nekada ih je potrebno i animirati kao deo neke grupe, na primer, ako modelujemo svetlo farova automobila koji se kreće. I kameru je, takođe, potrebno pozicionirati i usmeriti, te je moguće i nju dodati u graf scene. Na taj način moguće je i nju animirati, kao deo neke grupe, ako recimo želimo da zadamo pogled vozača automobila koji se kreće.

Renderovanje scene zadate u vidu grafa scene sastoji se u obilasku grafa scene počev od korena. Pritom dolazak u čvor objekta (primitive) podrazumeva njegovo iscrtavanje u skladu sa kumulativnom matricom transformacije koja na njega deluje, dok se obilaskom čvora transformacije tekuća kumulativna matrica ažurira tako što se njena tekuća vrednost množi s leve strane matricom transformacije

id čvora	akcija	T	stek
koren	$T = E$	E	-
T_0	$T = T \cdot T_0$	T_0	-
g_1	push(T)	T_0	T_0
T_1	$T = T \cdot T_1$	$T_0 \cdot T_1$	T_0
o_1	renderuj($T \cdot o_1$)	$T_0 \cdot T_1$	T_0
g_3	push(T)	$T_0 \cdot T_1$	$T_0, T_0 \cdot T_1$
T_5	$T = T \cdot T_5$	$T_0 \cdot T_1 \cdot T_5$	$T_0, T_0 \cdot T_1$
o_3	renderuj($T \cdot o_3$)	$T_0 \cdot T_1 \cdot T_5$	$T_0, T_0 \cdot T_1$
g_3	T=pop()	$T_0 \cdot T_1$	T_0
T_6	$T = T \cdot T_6$	$T_0 \cdot T_1 \cdot T_6$	T_0
o_4	renderuj($T \cdot o_4$)	$T_0 \cdot T_1 \cdot T_6$	T_0
g_1	T=pop()	T_0	-
T_2	$T = T \cdot T_2$	$T_0 \cdot T_2$	-
g_2	push(T)	$T_0 \cdot T_2$	$T_0 \cdot T_2$
T_3	$T = T \cdot T_3$	$T_0 \cdot T_2 \cdot T_3$	$T_0 \cdot T_2$
o_2	renderuj($T \cdot o_2$)	$T_0 \cdot T_2 \cdot T_3$	$T_0 \cdot T_2$
g_2	T=pop()	$T_0 \cdot T_2$	-
T_4	$T = T \cdot T_4$	$T_0 \cdot T_2 \cdot T_4$	-
g_3	push(T)	$T_0 \cdot T_2 \cdot T_4$	$T_0 \cdot T_2 \cdot T_4$
T_5	$T = T \cdot T_5$	$T_0 \cdot T_2 \cdot T_4 \cdot T_5$	$T_0 \cdot T_2 \cdot T_4$
o_3	renderuj($T \cdot o_3$)	$T_0 \cdot T_2 \cdot T_4 \cdot T_5$	$T_0 \cdot T_2 \cdot T_4$
g_3	T=pop()	$T_0 \cdot T_2 \cdot T_4$	-
T_6	$T = T \cdot T_6$	$T_0 \cdot T_2 \cdot T_4 \cdot T_6$	-
o_4	renderuj($T \cdot o_4$)	$T_0 \cdot T_2 \cdot T_4 \cdot T_6$	-

Tabela 3.1: Renderovanje grafa scene: za svaki čvor grafa scene koji se obilazi ispisana je akcija koja se izvršava, kumulativna matrica transformacije T i sadržaj steka transformacija.

sadržane u tom čvoru. Za održavanje kumulativne matrice transformacije koristi se stek transformacija. Prilikom obilaska grafa i nailaska na čvor grupisanja koji odgovara podobjektu, koji dalje hijerarhijski modelujemo i u kome se vrši grananje, na stek transformacija postavlja se tekuća matrica transformacije. Nakon obilaska jedne grane i povratka u taj čvor, kumulativna matrica transformacije se postavlja na vrednost sa vrha steka transformacija, a zatim se sa steka uklanja element sa vrha; na taj način vrši se povratak vrednosti kumulativne matrice transformacije na vrednost koja je bila prilikom ulaska u taj čvor. Obilaskom čvora svojstava postavlja se tekuće svojstvo objekta, npr. boja objekta se postavlja na zelenu. Ukoliko graf scene nema oblik stabla, on se ipak obilazi kao da je stablo. To će naime dovesti do toga da se u neke čvorove ulazi veći broj puta, ali to i jeste potrebno kada je jednu grupu potrebno veći broj puta iskoristiti i iscrtati.

Primer 3.11. U tabeli 3.1 prikazano je korak po korak kako bi izgledalo renderovanje grafa scene prikazanog na slici 3.23.

3.5 Pitanja

3.1 Ako je linearna transformacija u ravni data matricom transformacije T, a $\vec{e_1}$ i $\vec{e_2}$ predstavljaju standardnu bazu vektorskog prostora, čemu su jednake vrednosti $T(\vec{e_1})$ i $T(\vec{e_2})$? Kako nam ovo može poslužiti za izvođenje matrice linearne transformacije T?

3.2 Čemu je jednaka matrica skaliranja u ravni u odnosu na koordinatni početak? Da li se skaliranjem čuvaju uglovi između pravih u ravni?

3.3 Kako izgleda matrica rotacije u ravni oko koordinatnog početka za ugao ϕ? Na koji način je možemo dobiti?

3.4 Kako se u terminima homogenih koordinata zadaje tačka u ravni?

3.5 Kako se naziva tačka sa homogenim koordinatama $(x, y, 0)$?

3.6 Čemu u Dekartovom trodimenzionom prostoru pripadaju sve homogenizovane koordinate tačaka u ravni?

3.7 Kako u homogenim koordinatama predstaviti naredne transformacije: skaliranje u odnosu na koordinatni početak, rotaciju u odnosu na koordinatni početak, smicanje, translaciju?

3.8 Koja svojstva čuvaju afine transformacije?

3.9 Koja je transformacija inverzna skaliranju/translaciji/rotaciji/smicanju?

3.10 Šta je kompozicija više skaliranja/translacija/rotacija/smicanja?

3.11 Kog je oblika matrica izometrijskih transformacija?

3.12 Koje transformacije od pomenutih jesu izometrijske? (a) rotacija (b) skaliranje (c) translacija (d) smicanje

3.13 Za koje izometrijske transformacije kažemo da su direktne, a za koje da su indirektne?

3.14 Kako se može predstaviti svaka afina transformacija u ravni? Kako se može predstaviti svaka afina transformacija u trodimenzionom prostoru?

3.15 Izvesti matricu skaliranja u trodimenzionom prostoru sa faktorima skaliranja 2, -3 i $1/3$ u odnosu na tačku sa koordinatama $(1, -1, 3)$ i matricu njoj inverzne transformacije. Nije neophodno množiti matrice.

3.16 Izvesti opšti oblik matrice smicanja u trodimenzionom prostoru u pravcu Oyz koordinatne ravni.

3.17 Dokazati da rotacija i skaliranje u ravni komutiraju ako je $s_x = s_y$ ili je $\Theta = n\pi$ za celobrojno n, a inače ne komutiraju.

3.18 Data je jedinična kocka sa donjim levim temenom u tački $(0, 0, 0)$ i gornjim desnim u tački $(1, 1, 1)$. Izvesti matricu transformacije potrebne za rotiranje kocke za Θ stepeni oko glavne dijagonale u smeru suprotnom od smera kazaljke na časovniku kada se gleda duž dijagonale ka koordinatnom početku.

3.19 Kako odrediti sliku prave pri nekoj transformaciji? Kako odrediti sliku ravni pri nekoj transformaciji?

3.20 U kojim situacijama je potrebno izvršiti promenu koordinatnog sistema?

3.21 Od čega se sastoji graf scene?

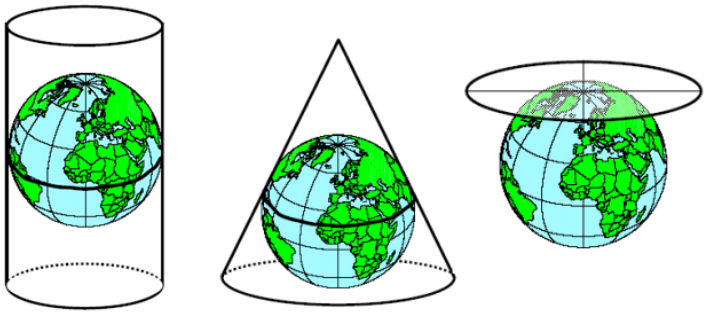
3.22 Čemu služe čvorovi grupisanja u grafu scene?

3.23 Nacrtati graf scene za model kućice koja se sastoji od kocke čija je stranica dužine 2 na koju je postavljena piramida iste osnove čiji se donji levi ugao nalazi u tački sa koordinatama $(2, 2)$.

Glava 4

Projektovanje

Prikazivanje trodimenzionih objekata na uređajima za prikaz koji su dvodimenzioni ostvaruje se posredstvom projekcija: objekti se prikazuju tako što se prikazuje njihova projekcija na ravan. Preciznije, objekti iz trodimenzionog sveta se odsecaju u odnosu na *trodimenzionu zapreminu pogleda* (eng. 3D view volume), a zatim projektuju na ravan. Sadržaj projekcije zapremine pogleda u ravni projekcije se dalje transformiše u oblast za prikaz.



Slika 4.1: Tipovi kartografskih projekcija: cilindrična, konusna i azimutalna.

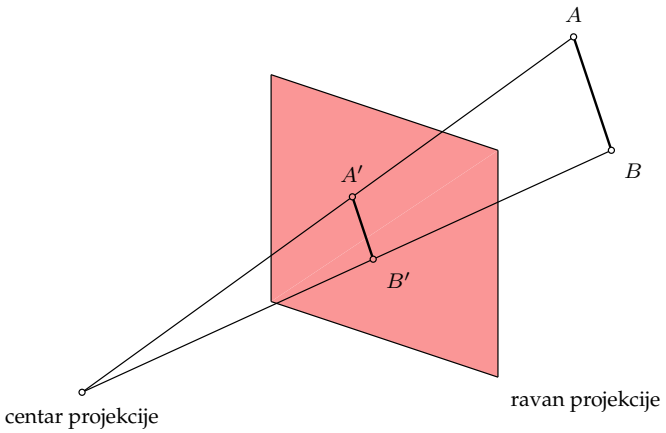
U opštem slučaju, *projekcija* (eng. projection) je preslikavanje iz koordinatnog sistema dimenzije n u koordinatni sistem dimenzije manje od n. Projekcije kojima se vrši preslikavanje u ravan nazivaju se *planarne projekcije* (eng. planar projection). Pomenimo da postoje i projekcije koje nisu planarne kao, na primer, kartografska projekcija (slika 4.1), ali se njima nećemo baviti u ovom materijalu[1].

Od posebnog interesa za računarsku grafiku su projekcije iz trodimenzionog prostora u ravan. One su određene *centrom projekcije* (eng. centre of projection) i *ravni projekcije* (eng. projection plane). U zavisnosti od toga da li se centar projekcije nalazi na konačnom rastojanju od ravni projekcije ili ne, projekcije se dele na:

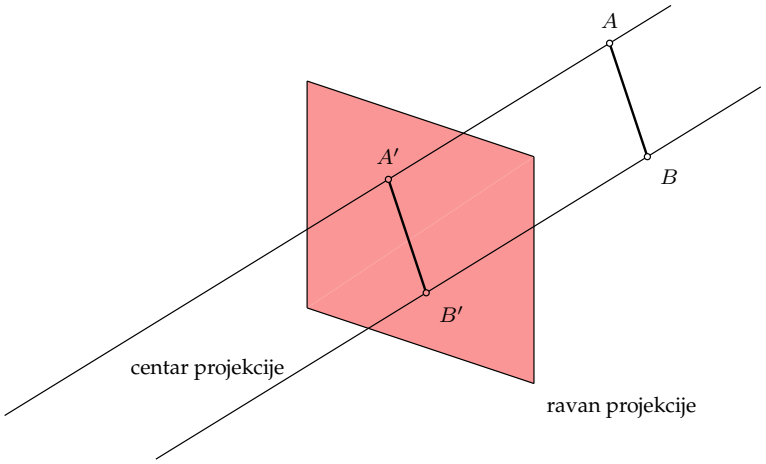
- *perspektivne* (eng. perspective projection) – kada je centar projekcije na konačnom rastojanju od ravni projekcije (slika 4.2),
- *paralelne* (eng. parallel projection) – kada je centar projekcije na beskonačnom rastojanju od ravni projekcije (slika 4.3).

Paralelna projekcija je dobila naziv po tome što je centar projekcije na beskonačnom rastojanju od ravni projekcije, te su zraci projekcije međusobno paralelni. Prilikom zadavanja perspektivne projekcije potrebno je eksplicitno zadati centar projekcije koji odgovara poziciji kamere (odnosno oku posmatrača) čime se simulira šta je ono što vidi kamera (odnosno oko posmatrača) dok je za paralelnu projekciju potrebno zadati *pravac projekcije* (eng. direction of projection) jer u ovom slučaju prave ne konvergiraju ka kameri (odnosno oku posmatrača). Centar projekcije se kod perspektivne projekcije nekada naziva i *tačka pogleda* (eng. view point), jer označava lokaciju kamere (odnosno oka posmatrača).

[1] Kartografske projekcije predstavljaju način prikazivanja Zemljine površine u ravni. Jedan način da se to postigne jeste da se Zemljina površina projektuje na površ koju je moguće razviti u ravan bez rastezanja, cepanja i sažimanja (to može biti omotač valjka ili kupe ili ravan), a da se dobijena projekcija nakon toga razvije u ravan.



Slika 4.2: Perspektivna projekcija.



Slika 4.3: Paralelna projekcija.

Centar projekcije kod perspektivnog projektovanja možemo predstaviti tačkom čije su homogene koordinate $(x, y, z, 1)$. Centar projekcije kod paralelne projekcije možemo predstaviti beskonačno dalekom tačkom sa koordinatama $(x, y, z, 0)$, a kao što već ranije napomenuli ovim koordinatama je određen vektor u trodimenzionom prostoru koji odgovara pravcu projektovanja. Dakle, pravcu projekcije kod paralelne projekcije odgovara beskonačno daleka tačka, pa paralelnu projekciju možemo shvatiti kao perspektivnu projekciju čiji je centar beskonačno daleka tačka.

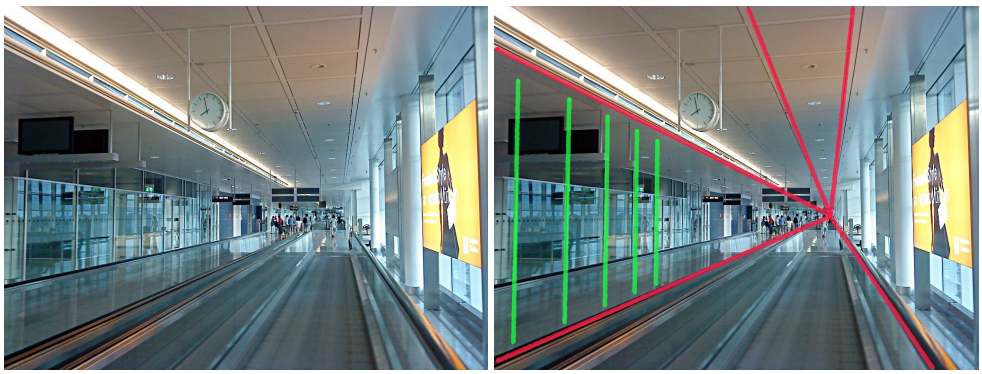
Vizualni efekat perspektivne projekcije odgovara čovekovom vizualnom sistemu, kao i fotografskim sistemima. Njega karakteriše tzv. *perspektivno skraćenje* (eng. perspective foreshortening): veličina perspektivne projekcije objekta obrnuto je srazmerna rastojanju tog objekta od centra projekcije. Stoga, iako slika u ovoj projekciji izgleda realistično, njen praktičan značaj za određivanje tačnih oblika i dimenzija predstavljenih objekata je beznačajan. Paralelna projekcija daje manje realističnu sliku zbog nedostatka perspektivnog skraćenja. Međutim, određivanje oblika i dimenzija predstavljenih objekata je kod paralelne projekcije dosta jednostavnije. Paralelne prave se pri paralelnom projektovanju preslikavaju u paralelne prave, dok se kod perspektivnog projektovanja paralelne prave ne preslikavaju uvek u paralelne prave. I kod paralelne i kod perspektivne projekcije uglovi se čuvaju samo na ravnima koje su paralelne sa ravni projekcije.

U nastavku ovog poglavlja ćemo detaljnije razmotriti svaku od ove dve vrste projekcije.

4.1 Perspektivna projekcija

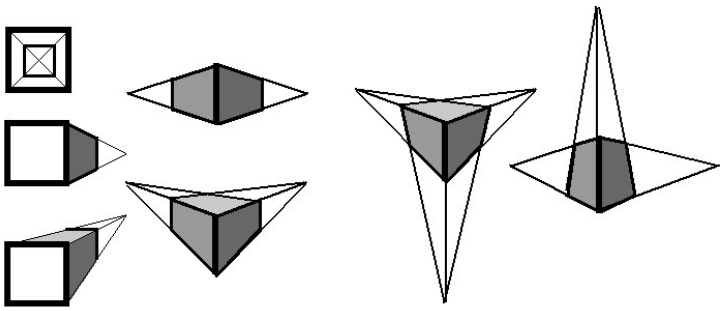
Čovekov vizualni sistem karakteriše upotreba perspektivne projekcije: objekti koji su dalji od nas deluju nam manji od onih koji su nam bliži, odnosno veličina objekta se smanjuje sa povećanjem rastojanja od centra projekcije. Pritom, skraćenje nije uniformno, odnosno međusobno jednaka rastojanja na pravoj ne projektuju se nužno u jednaka. Dodatno, međusobno paralelne prave ne moraju ostati paralelne na slici.

Preciznije, kod perspektivne projekcije paralelne prave koje su paralelne ravni projektovanja slikaju se u paralelne prave, dok se paralelne prave koje nisu paralelne ravni projekcije seku na slici (slika 4.4). Prednost perspektivne u odnosu na paralelnu projekciju je u tome što daje realističan prikaz i osećaj trodimenzionalnosti objekta. Međutim, perspektivnim projektovanjem se ne čuva ni oblik objekta, niti odnosi (osim kada objekat seče ravan projekcije).



Slika 4.4: Fotografija minhenskog aerodroma: paralelne prave koje su paralelne ravni projekcije ostaju paralelne (duži prikazane zelenom bojom), dok se paralelne prave koje nisu paralelne ravni projekcije seku u tački nedogleda (duži prikazane crvenom bojom).

Perspektivne projekcije proizvoljnog skupa paralelnih pravih koje nisu paralelne ravni projekcije seku se u tzv. *tački nedogleda* (eng. vanishing point). U trodimenzionom prostoru možemo reći da se paralelne prave seku u beskonačno dalekoj tački, te se na tačku nedogleda može gledati kao na perspektivnu projekciju beskonačno daleke tačke.



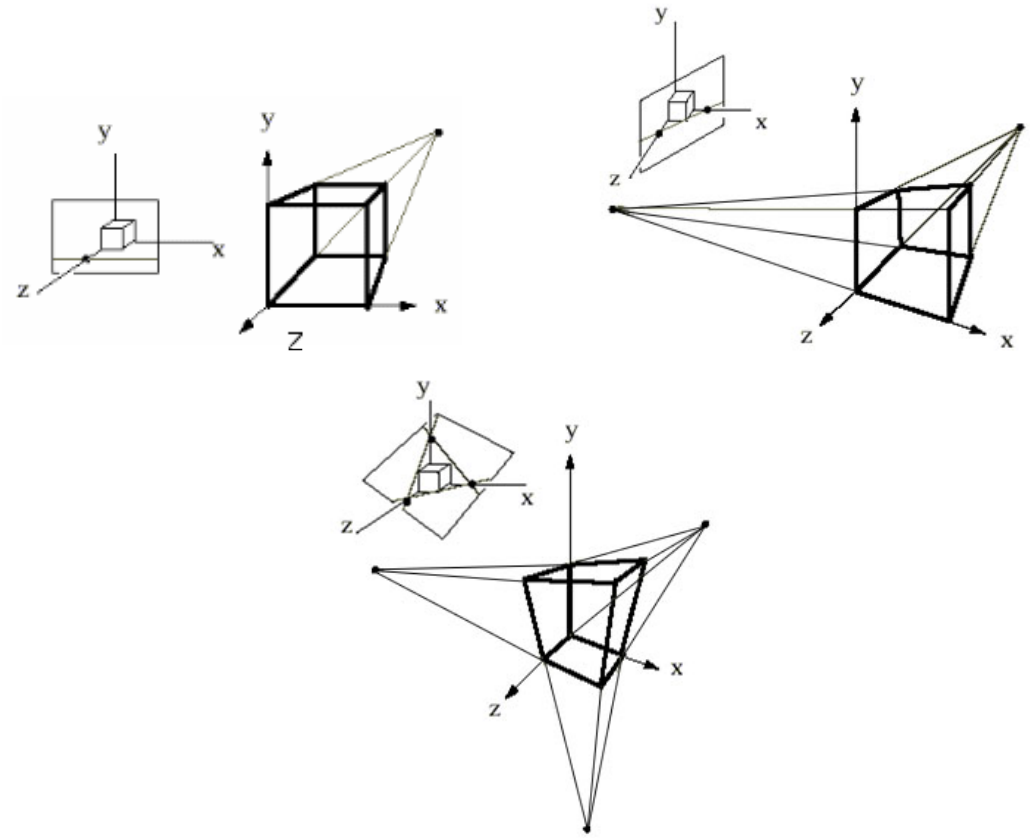
Slika 4.5: Perspektivna projekcija kocke sa jednom, dve i tri osne tačke nedogleda. Perspektivne projekcije sa istim brojem tačaka nedogleda se međusobno razlikuju po tome da li se objekat koji posmatramo nalazi u visini očiju posmatrača (čovečja perspektiva), ispod visine očiju posmatrača (ptičja perspektiva) ili iznad visine očiju (žablja perspektiva).

Ako je skup paralelnih pravih paralelan sa koordinatnom osom, onda njenu tačku nedogleda nazivamo *osna tačka nedogleda* (eng. axis vanishing point). Pošto u trodimenzionom prostoru imamo tri koordinatne ose, perspektivna projekcija može imati najviše tri osne tačke nedogleda. Štaviše, perspektivne projekcije se najčešće i dele na osnovu broja osnih tačaka nedogleda pa razlikujemo perspektivnu projekciju sa jednom, dve ili tri osne tačke nedogleda (slika 4.5). U praksi se najčešće koriste perspektivne projekcije sa jednom i dve tačke nedogleda.

Za pravougaone figure čije su normale strana kolinearne sa koordinatnim osama x, y i z, broj tačaka nedogleda jednak je broju koordinatnih osa koje presecaju ravan projekcije (slika 4.6).

4.1.1 Istorijat korišćenja perspektive u umetnosti

Perspektivna projekcija se u umetnosti koristi za dobijanje realističnih slika. Međutim, koncept perspektive kakav mi danas uzimamo „zdravo za gotovo“ smatra se relativno novim otkrićem u istoriji umetnosti. Veruje se da su Grci i Rimljani razumeli perspektivu, ali da se tokom vremena to znanje izgubilo. Nakon toga pa sve do četraestog veka bilo je veoma malo pokušaja da se realistično dočara svet kakav nas okružuje i kakav smo navikli da vidimo.



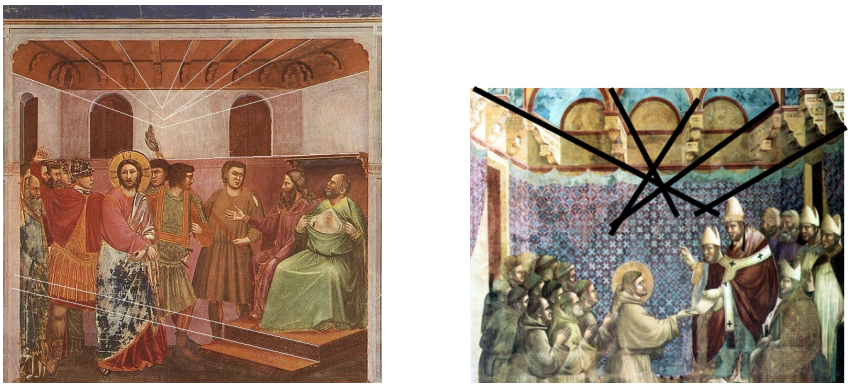
Slika 4.6: Perpsektivna projekcija sa jednom, dve i tri osne tačke nedogleda kocke čije su normale strana kolinearne koordinatnim osama (preuzeto iz kursa U Brown).

U starom Egiptu veličina i pozicija likova na slici zavisila je od njihovog značaja, a ne od njihove prave veličine i rastojanja do posmatrača (slika 4.7 a)). Srednjevekovno slikarstvo se dominantno svodilo na predstavljanje religioznih tema, a ne čovekovog iskustva i nije se težilo tome da se kreira iluzija dubine i prostora na slikama. I tada su na slikama kao najveće prikazivane najznačajnije figure i nije se vodilo računa o njihovom rastojanju od posmatrača (slika 4.7 b)).

Slika 4.7: (a) Egipatska zidna umetnost. (b) Srednjevekovno slikarstvo.

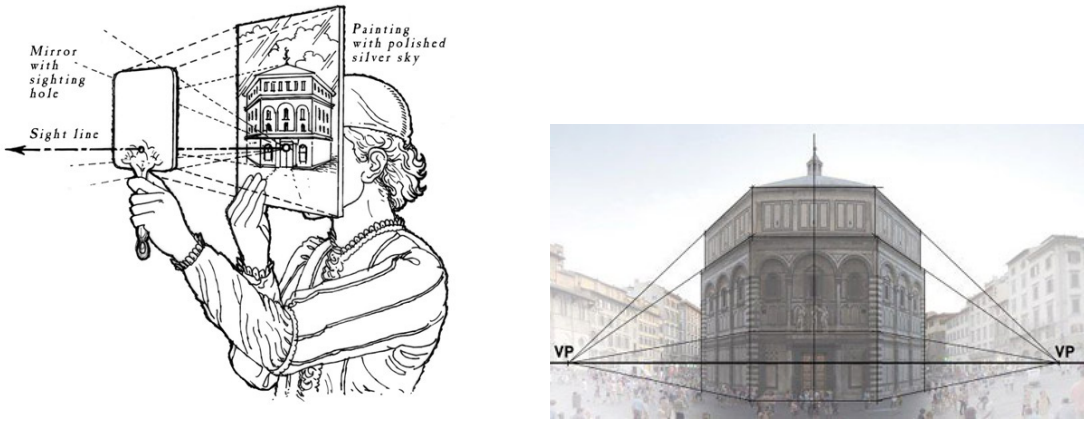
Italijanski slikar Đoto jedan je od prvih umetnika koji je pokušao da dočara iluziju dubine. Često se baš njemu pripisuje zasluga da je uveo ranu formu perspektive u umetnost. Đotova slika „Scene iz Hristovog života“ (slika 4.8 a)), naslikana sto godina pre Bruneleskijeve demonstracije perspektive, predstavlja jedan od prvih primera primene konvergencija paralelnih pravih u domenu perspektive. Iako je perspektiva na slici i dalje daleko od vida perspektive na koji smo danas navikli i paralelne prave ne konvergiraju ka jedinstvenoj tački nedogleda (slika 4.8 b)), na njegovim slikama je prisutna ideja da se sa povećanjem rastojanja od posmatača prave koje se nalaze iznad nivoa oka posmatrača

spuštaju, dok se prave ispod nivoa oka posmatrača izdižu.



Slika 4.8: (a) „Scene iz Hristovog života“, Đoto, 1305. (b) „Legenda sv. Franje“, Đoto, 1297.

Linearna perspektiva, kao način stvaranja iluzije prostora i dubine na slici, javila se iz želje da se što vernije predstave eksterijer i enterijer različitih građevina. Ona predstavlja pojavu da paralelne prave konvergiraju ka jedinstvenoj tački nedogleda[2]. Jedan od prvih primera upotrebe linearne perspektive jeste slika prednje strane katedrale u Firenci kreirana od strane italijanskog arhitekte Filipa Bruneleskija (Filippo Brunelleschi) 1415. godine (slika 4.9). Bruneleski je realističnost dobijenog prikaza demonstrirao narednim eksperimentom: u sredini slike u visini posmatrača probušio je rupu i u jednoj ruci držao je sliku okrenutu poleđinom ka sebi, a u drugoj ogledalo upereno ka slici tako da se kroz otvor posredstvom ogledala mogla videti slika crkve. Naizmeničnim podizanjem i spuštanjem ogledala došao je do zaključka da nacrtana slika verno odgovara načinu na koji covek vidi fasadu katedrale.



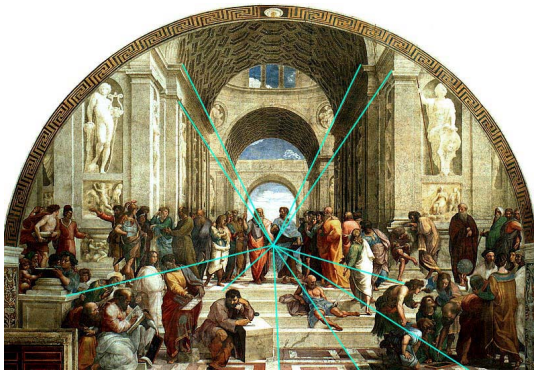
Slika 4.9: Proces generisanja slike koja sadrži linearnu perspektivu, Bruneleski, 1415.

Nakon toga, drugi autori krenuli su da uvode nove vidove perspektive na kojima su se građevine i predeli realistično smanjivali sa porastom udaljenja od posmatrača. Uvođenje koncepta perspektive u sferu umetnosti drastično je doprinelo realističnosti slike: ona je omogućila da odnosi veličina na slici budu odgovarajući, da udaljeniji objekti deluju manji, i slično.

Do kraja petnaestog veka, umetnici su u potpunosti ovladali perspektivom i imali su znanja koja su im omogućavala da kreiraju veoma realistične slike. Kao jedan od najpotpunijih primera upotrebe perspektive često se navodi Rafaelova slika „Atinska škola“ (slika 4.10). Ova slika se smatra izvanrednim primerom korišćenja perspektive u cilju naglašavanja kompozicije slike. Na slici 4.11 prikazana je studija za sliku „Poklonjenje mudraca“ Leonarda da Vinčija sa koje se može zaključiti da su umetnici veoma ozbiljno pristupali problemu kreiranja realističnosti prikaza na slikama.

Albreht Direr je 1525. godine napravio duborez kojim je ilustrovao način na koji je moguće kreirati perspektivnu sliku proizvoljnog objekta (slika 4.12). Duborez prikazuje dva muškarca koji prave sliku laute korišćenjem aparature sačinjene od strune, pravougaonog rama sa poklopcem i olovke. Jedan

[2]Pored linearne, postoji i atmosferska perspektiva koja se postiže tako što se udaljeniji objekti slikaju kao manje vidljivi od onih koji su bliže posmatraču.



Slika 4.10: „Atinska škola“, Rafael, 1511.

Slika 4.11: Leonardo da Vinči: studija za Poklonjenje mudraca i konačna (nezavršena) slika (oko 1481).

kraj dugačke strune (u levom uglu slike) zakačen je za vrh pokazivača, dok je drugi pričvršćen za zid i završava se malim tegom čiji je smisao da održava napetost strune. Zadatak prvog muškarca na slici je da pomera pokazivač tako da dodiruje različite tačke na objektu čiju perspektivnu sliku želimo da dobijemo. Pored strune, aparatura sadrži pravougaoni ram sa poklopcem, koji je šarkama povezan sa ramom, i koji omogućava otvaranje i zatvaranje rama. Na poklopac rama zakačen je papir na kome treba nacrtati sliku. Struna prolazi kroz ram i na mestu gde prođe kroz ram drugi muškarac postavlja olovku. Struna se zatim pomera u stranu, poklopac rama se zatvara i olovka ostavlja trag na odgovarajućem mestu na papiru. Ovaj postupak se ponavlja sve dok se ne iscrta kompletna slika. Dobijena slika odgovara perspektivnom pogledu na lautu kada bi se posmatrač nalazio u tački gde je struna pričvršena za zid.

Početkom dvadesetog veka umetnici su krenuli da eksperimentišu i da ignorišu tradicionalna pravila perspektive, što je dovelo do razvoja pravaca kao što su impresionizam, kubizam i apstraktna umetnost.

4.2 Paralelna projekcija

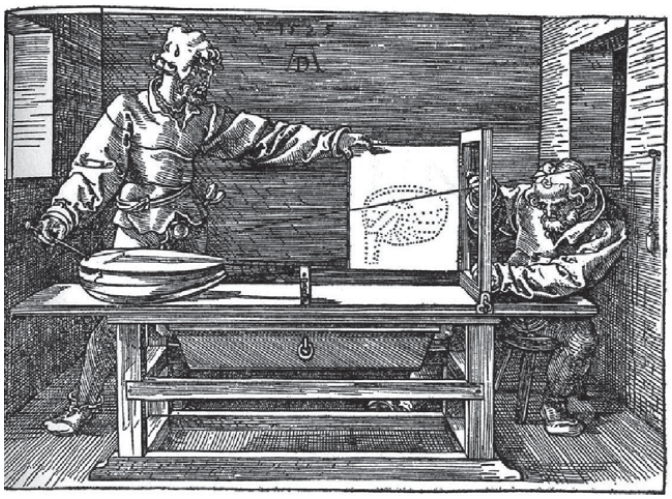
Kao što smo već pomenuli, kod paralelnog projektovanja centar projekcije je beskonačno daleka tačka te su svi zraci projektovanja međusobno paralelni.

U opštem slučaju paralelna projekcija određena je uglom koji pravac projektovanja zahvata sa ravni projekcije i položajem ravni projekcije u odnosu na glavne ose objekta (odgovarajući ugao i pozicija). U daljem tekstu pretpostavićemo da je koordinatni sistem koji razmatramo vezan za objekat i da su glavne ose objekta upravne na koordinatne ose.

Paralelnu projekciju kod koje je pravac projektovanja jednak (ili suprotan) vektoru normale ravni projektovanja nazivamo *ortogonalnom projekcijom* (eng. ortographic projection), dok paralelnu projekciju kod koje to nije slučaj nazivamo *kosom projekcijom* (eng. oblique projection).

4.2.1 Ortogonalna projekcija

Ortogonalna projekcija je paralelna projekcija kod koje su projektivni zraci upravni na ravan projektovanja. U zavisnosti od odnosa projekcione ravni i koordinatnih osa, razlikujemo dve vrste ortogonal-



Slika 4.12: Direrov duborez koji prikazuje dva čoveka koji prave sliku laute korišćenjem aparature koja odgovara današnjem procesu renderovanja slike, 1525.

nih projekcija:

- *ortografske* projekcije (eng. ortographic or multiview projection) – kod njih je projekciona ravan upravna na neku od koordinatnih osa,
- *aksonometrijske* projekcije (eng. axonometric projection) – kod njih projekciona ravan nije upravna ni na jednu od koordinatnih osa.

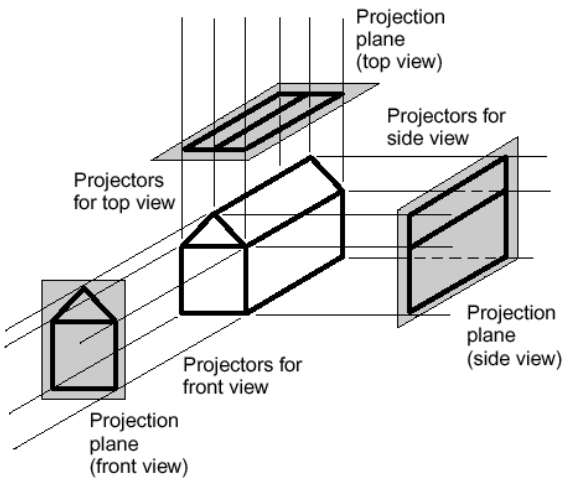
U zavisnosti od toga kojoj strani objekta je ravan projekcije paralelna, razlikujemo tri vrste ortografskih projekcija: *pogled spreda* (eng. front-elevation), *pogled odozgo* (eng. top-elevation) i *pogled sa strane* (eng. side-elevation) (slika 4.13). Svaki od ovih pogleda predstavlja samo dve dimenzije objekta, te je iz pogleda spreda moguće očitati širinu i visinu objekta, iz pogleda odozgo širinu i dubinu objekta, a iz pogleda sa strane dubinu i visinu objekta. Odabir pogleda spreda određuje u potpunosti i druga dva pogleda na objekat.

S obzirom na to da one verno prikazuju jednu stranu objekta, ortografske projekcije se često koriste u inženjerskim i arhitektonskim crtežima za prikazivanje delova mašina i građevina jer se sa njih mogu izmeriti rastojanja i uglovi. Iz ovog razloga pogodne su i za prikaz dizajna delova koji će se proizvoditi na udaljenoj lokaciji. Međutim, svaka od projekcija prikazuje samo jednu stranu objekta, te se ovim tipom projektovanja ne stiče utisak trodimenzionalnosti objekta i potrebno je koristiti više različitih pogleda kako bi se stekao uvid u ceo objekat. Uobičajeno su svi pogledi dati u istoj srazmeri (duž je prikazana u istoj dužini u svim pogledima).

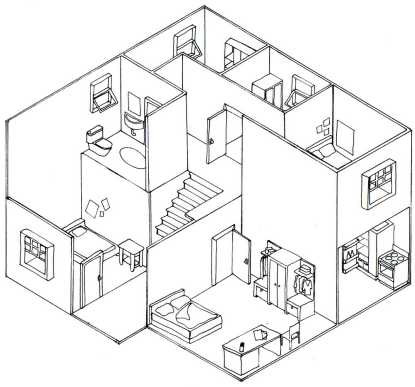
Aksonometrijska projekcija je ortogonalna projekcija koja koristi ravan projekcije koja nije upravna ni na jednu od koordinatnih osa. Stoga se u ovoj projekciji prikazuje veći broj strana objekta odjednom i za razliku od ortografske projekcije dobija se utisak trodimenzionalnosti objekta (slika 4.14). Međutim, za razliku od perspektivne projekcije, kod aksonometrijske projekcije skraćenje je uniformno i na njega ne utiče rastojanje od centra projekcije. Aksonometrijsko projektovanje čuva paralelnost pravih, ali ne i uglove, dok se rastojanja mogu meriti duž svake od glavnih osa objekta, u opštem slučaju sa različitim faktorima skaliranja.

Kod aksonometrijskog projektovanja ravan projekcije seče sve tri ose koordinatnog sistema u kom je definisan objekat. Na osnovu uglova koji normala ravni projekcije zahvata sa koordinatnim osama razlikujemo:

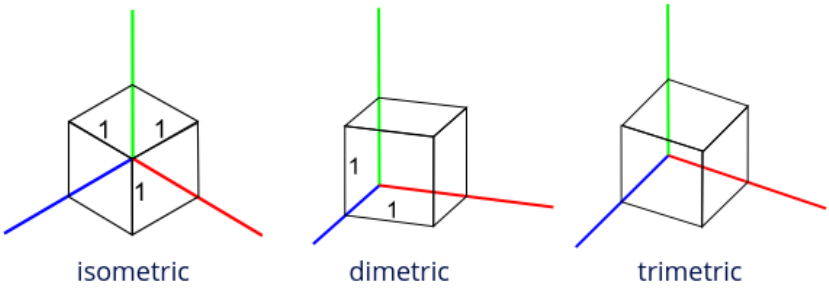
- *izometrijsku projekciju* (eng. isometric projection) kod koje normala ravni projekcije zahvata jednake uglove sa svim trima koordinatnim osama,
- *dimetrijsku projekciju* (eng. dimetric projection) kod koje je ugao između normale ravni projekcije i neke dve koordinatne ose isti,
- *trimetrijsku projekciju* (eng. trimetric projection) kod koje su sva tri ugla između normale ravni projekcije i koordinatnih osa međusobno različita.



Slika 4.13: Ortografska projekcija modela kuće: pogled spreda, odozgo i sa strane (preuzeto iz kursa U Brown).



Slika 4.14: Aksonometrijska projekcija enterijera kuće.



Slika 4.15: Izometrijska, dimetrijska i trimetrijska projekcija kocke.

Na slici 4.15 prikazana je izometrijska, dimetrijska i trimetrijska projekcija jedinične kocke. Primetimo da su u slučaju izometrijske projekcije sve stranice kocke iste dužine, u slučaju dimetrijske su projekcije vertikalnih i horizontalnih stranica kocke istih dužina, a bočne stranice nisu, dok su kod trimetrijske projekcije dužine projekcija vertikalnih, horizontalnih i bočnih stranica kocke međusobno različite. Važno svojstvo izometrijske projekcije jeste to da se dužine duž svake od tri koordinatne ose jednako skraćuju, te su u ovoj projekciji sve stranice kocke iste dužine. Ovo svojstvo omogućava da se vrše merenja u smeru sve tri ose sa istim faktorom skaliranja (otud i naziv izometrijska od grčkog termina „jednaka mera“). Dodatno, koordinatne ose se projektuju na takav način da grade jedna sa drugom isti ugao (od 120°).

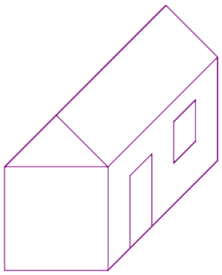
Izometrijsko projektovanje se koristi za ilustracije po katalozima, za objašnjenje mehaničkih sistema (npr. u uputstvima za sklapanje Lego kocki), kao i za 3D modelovanje u realnom vremenu (Maya, AutoCad, . . .)

Slika 4.16: Primer korišćenja aksonometrijskih projekcija u računarskim igrama.

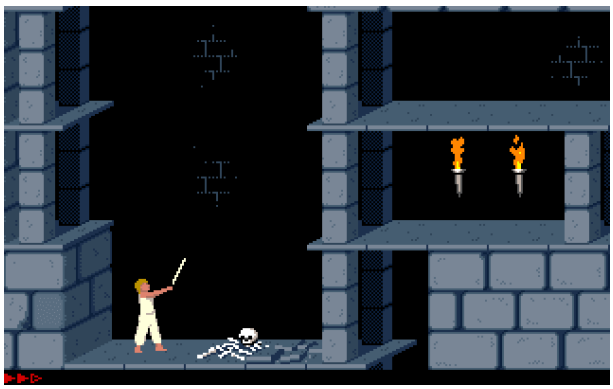
Aksonometrijske projekcije (pre svega izometrijske i dimetrijske projekcije) su godinama korišćene u video igrama kao tehnika koja pruža neki vid trodimenzionalnosti prikaza koju je bilo moguće postići sa ograničenim resursima (slika 4.16). I danas se koriste kada je cilj da se objekti u daljini vide jednako dobro kao i bliži objekti.

4.2.2 Kosa projekcija

Kosa projekcija je paralelna projekcija kod koje pravac projekcije nije jednak (niti suprotan) normali ravni projekcije, dok je ravan projekcije najčešće upravna na neku od koordinatnih osa, odnosno paralelna je jednoj strani objekta (slika 4.17). Kosom projekcijom se može predstaviti tačan oblik jedne strane objekta, te se, ako je potrebno, mogu izvesti precizna merenja te strane. Nedostatak perspektivnog skraćenja olakšava poređenje veličina, međutim, iako se kod kose projekcije donekle dobija utisak trodimenzionalnosti objekta, baš taj nedostatak perspektivnog skraćenja čini prikaz nedovoljno realističnim. Kosa projekcija se, uz izometrijsku projekciju, često koristi za tehničke crteže i u video igrama (4.18).



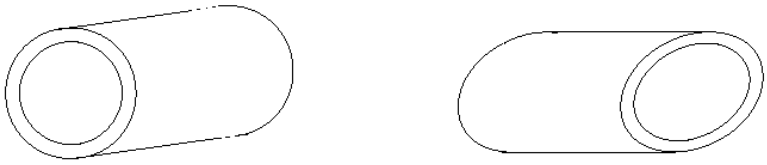
Slika 4.17: Kosa projekcija modela kuće.



Slika 4.18: Korišćenje kabinet kose projekcije u video igri Prince of Persia (preuzeto sa https://medium.com/retronator-magazine).

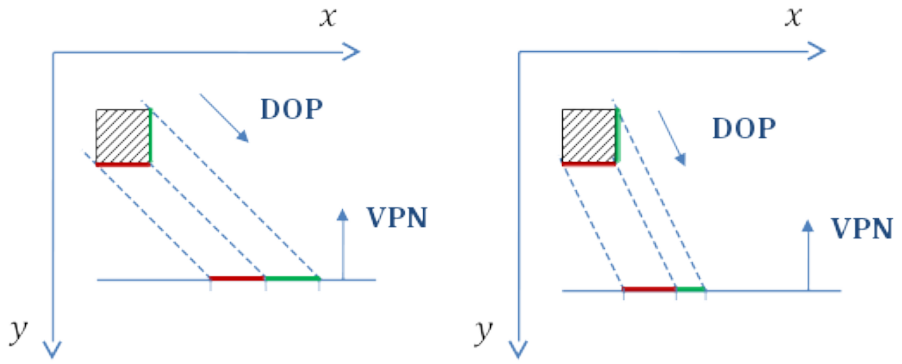
Kod kose projekcije jako je važan odabir ravni projektovanja jer ukoliko se on ne izabere pažljivo objekti mogu da izgledaju iskrivljeno (na primer, krugovi postaju elipse, slika 4.19). Stoga se često ravan

projekcije bira tako da bude paralelna strani objekta koja je najmanje pravilna, odnosno onoj strani koja sadrži najveći broj zakrivljenih površina.



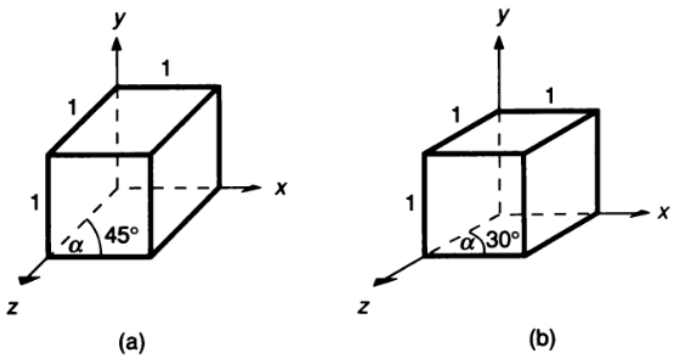
Slika 4.19: Kosa projekcija valjka u slučaju a) kada je ravan projekcije paralelna kružnoj strani objekta, b) kada ravan projekcije nije paralelna kružnoj strani objekta (preuzeto sa kursa U Brown).

Najčešće korišćeni tipovi kosih projekcija su *cavalier* i *cabinet* projektovanje. Cavalier projektovanje[3] koristi pravac projektovanja koji zahvata ugao od $45°$ sa ravni projektovanja (slika 4.20 a)). Kao rezultat, projekcije duži upravnih na ravan projekcije imaju iste dužine kao i same duži (tj. za njih nema skraćenja).



Slika 4.20: Najčešće korišćeni tipovi kosih projekcija: a) cavalier projektovanje b) cabinet projektovanje (oznake na slici: DOP (eng. direction of projection) - pravac projektovanja, VPN (eng. view plane normal) - normala ravni projektovanja).

Cavalier projekcije se međusobno mogu razlikovati prema veličini ugla koji zahvataju projekcije duži upravne na ravan projektovanja sa horizontalnom koordinatnom osom. Taj ugao je obično $45°$ ili $30°$. Razmotrimo dve cavalier projekcije kocke date na slici 4.21. Koordinatne ose x i y su upravne u projekciji, dok je z osa nacrtana dijagonalno i zahvata redom uglove od $45°$ ili $30°$ sa projekcijom x koordinatne ose. Primetimo da su u oba slučaja projekcije svih stranica kocke jedinične dužine.

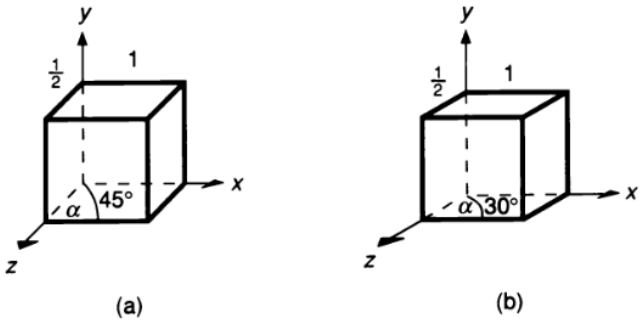


Slika 4.21: Cavalier projekcija jedinične kocke na ravan $z = 0$ kada projekcije duži upravnih na ravan projekcije sa x osom zahvataju ugao od (a) $45°$ (b) $30°$.

Za razliku od cavalier projektovanja, cabinet projektovanje[4] koristi pravac projektovanja koji zahvata ugao $\arctan 2 \approx 63.4°$ sa ravni projektovanja. Ova vrednost ugla odabrana je tako da projekcije duži

[3] Naziv cavalier projekcije potiče od francuske reči za konjanika i odgovara načinu na koji se objekti vide sa neke visoke tačke.
[4] Cabinet projekcija je svoj naziv dobila po engleskoj reči za orman jer se često koristi za ilustracije u industriji nameštaja.

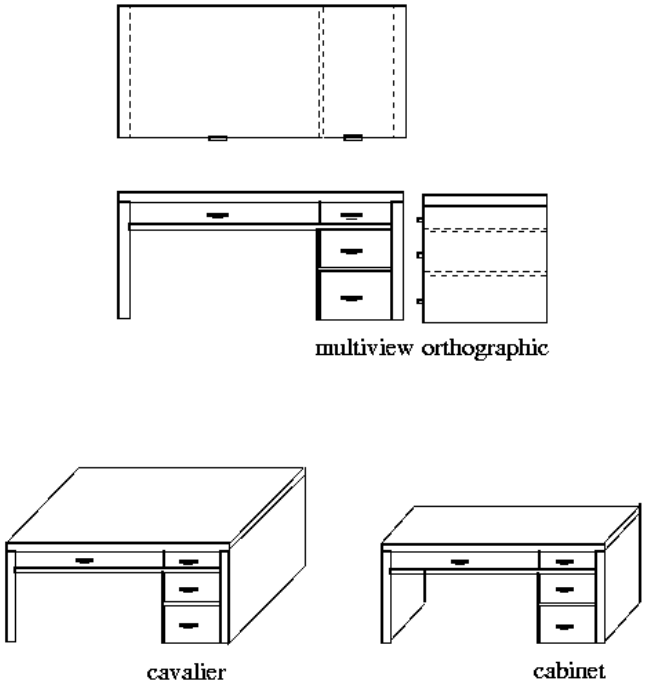
upravnih na ravan projekcije imaju dva puta manju dužinu u odnosu na same duži, čime se dobija realističniji prikaz u odnosu na cavalier projektovanje (slika 4.20 b)).



Slika 4.22: Cabinet projekcija jedinične kocke na ravan $z = 0$ kada projekcije duži upravnih na ravan projekcije sa x osom zahvataju ugao od (a) $45°$ (b) $30°$.

I cabinet projekcije se međusobno razlikuju prema veličini ugla koji zahvataju projekcije duži upravnih na ravan projektovanja sa x koordinatnom osom. Taj ugao je obično $45°$ ili $30°$ (slika 4.22). Primetimo da su projekcije stranica kocke paralelnih osama x i y jedinične dužine, dok su projekcije stranica paralelnih osi z duplo kraće.

Na slici 4.23 prikazan je radni sto korišćenjem različitih vrsta projekcija: ortografske projekcije putem sva tri pogleda (spreda, odozgo i sa strane), cavalier i cabinet projekcije.

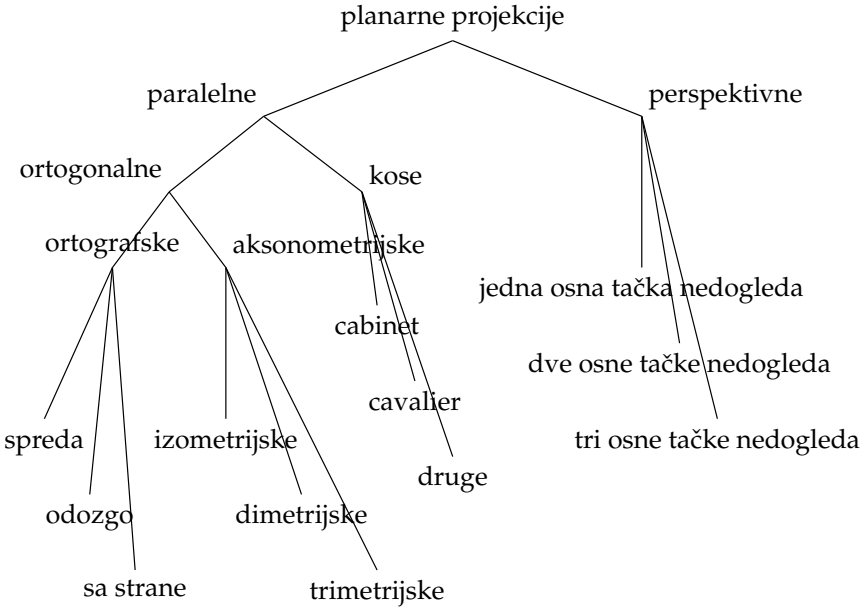


Slika 4.23: Projekcija radnog stola korišćenjem a) ortografskog projektovanja b) cavalier projektovanja c) cabinet projektovanja (preuzeto sa U Brown).

Klasifikacija svih razmatranih planarnih projekcija prikazana je na slici 4.24.

4.3 Primer izračunavanja projekcija tačaka

U ovom poglavlju razmotrićemo kako izračunati projekcije tačaka pri različitim vrstama projektovanja. Pokazaćemo da se svaka od razmatranih projekcija može zadati matricom dimenzije 4×4 u homogenim koordinatama, što omogućava njihovo kombinovanje sa drugim matricama transformacije. Pritom nemamo neku sistematičnu proceduru za izvođenje matrica projekcije (kao što je to recimo bilo kod izvođenja matrica linearnih transformacija), već ćemo probati da dođemo do jednačina za koordinate



Slika 4.24: Klasifikacija planarnih projekcija.

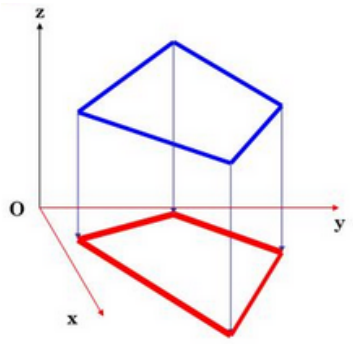
tačaka projekcije na osnovu koordinata polazne tačke i da na osnovu toga izvedemo odgovarajuću matricu. Sve ove matrice imaće determinantu 0, odnosno neće biti invertibilne jer će većem broju tačaka iz trodimenzionog prostora odgovarati ista tačka u projekciji.

Od svih razmatranih vrsta projektovanja, nama će biti najinteresantnija perspektivna projekcija.

4.3.1 Ortografsko projektovanje

Razmotrimo ortografsko projektovanje na ravan $z = 0$ i izvedimo matricu ovog projektovanja (slika 4.25). S obzirom na to da je pravac projekcije paralelan z koordinatnoj osi, za koordinate projekcije $P_p(x_p, y_p, z_p)$ tačke $P(x, y, z)$ važi:

$$x_p = x, \quad y_p = y, \quad z_p = 0$$



Slika 4.25: Ortografsko projektovanje na ravan Oxy.

Ova vrsta projektovanja se može zadati matricom:

$$M_{orth} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

a matrična jednačina odgovarajuće transformacije glasi:

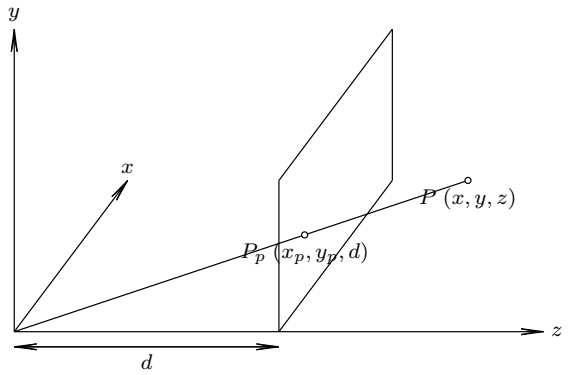
$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = M_{orth} \cdot P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix}$$

U opštem slučaju, ako bi ravan projektovanja imala jednačinu $z = d$, matrica ortografskog projektovanja bila bi:

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = M_{orth} \cdot P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix}$$

4.3.2 Perspektivno projektovanje

Razmotrimo primer perspektivnog projektovanja kod koga je centar projektovanja koordinatni početak, a ravan projekcije ravan $z = d$ (slika 4.26).



Slika 4.26: Perspektivno projektovanje na ravan $z = d$ sa centrom projekcije u koordinatnom početku.

Koordinate projekcije $P_p(x_p, y_p, z_p)$ tačke $P(x, y, z)$ mogu se izračunati na osnovu sličnosti trouglova:

$$\frac{x_p}{d} = \frac{x}{z} \quad \frac{y_p}{d} = \frac{y}{z}$$

odakle sledi

$$x_p = \frac{xd}{z} \quad y_p = \frac{yd}{z}$$

Kada bi dobijeni izrazi za x_p, y_p i z_p predstavljali linearnu kombinaciju vrednosti x, y, z i 1, bilo bi jednostavno doći do matrice transformacije, slaganjem koeficijenata po vrstama. Međutim, s obzirom na to da se u izrazima za x_p i y_p javlja z u imeniocu razlomka iskoristićemo to što matricu projektovanja zadajemo u homogenim koordinatama. Naime, umesto da homogenu koordinatu tačke P_p postavimo na 1, možemo joj postaviti neku drugu pogodnu vrednost, a onda na kraju izvršiti homogenizaciju koordinata. Primetimo da izraze za x_p i y_p možemo pogodnije zapisati:

$$x_p = \frac{x}{z/d} \quad y_p = \frac{y}{z/d}$$

I u izrazu za x_p i u izrazu za y_p se u imeniocu razlomka javlja izraz z/d, te homogenu koordinatu tačke P_p možemo postaviti na z/d. Pritom i vrednost z_p možemo predstaviti na sličan način:

$$z_p = \frac{z}{z/d}$$

Dakle, da bismo dobili perspektivnu projekciju date tačke $P(x, y, z, 1)$ potrebno je odrediti transformaciju koja je preslikava u tačku $P_p(x, y, z, z/d)$. Opisana transformacija može se predstaviti narednom

matricom:

$$M_{persp} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

a jednačina ovog perspektivnog projektovanja glasi:

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = M_{persp} \cdot P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \frac{z}{d} \end{bmatrix}$$

Dakle, Dekartove koordinate projekcije tačke (x, y, z) jednake su:

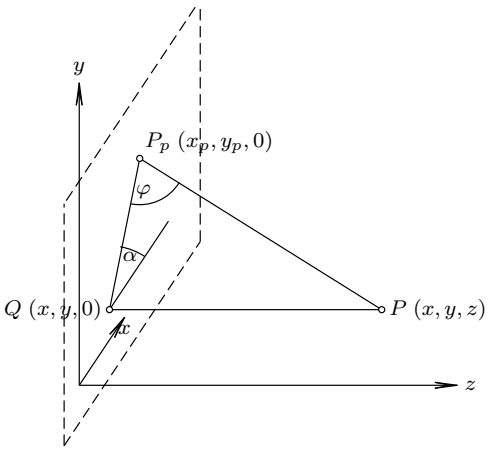
$$\left(\frac{X}{W}, \frac{Y}{W}, \frac{Z}{W}\right) = (x_p, y_p, z_p) = \left(\frac{x}{z/d}, \frac{y}{z/d}, d\right)$$

što je i trebalo dobiti.

Deljenje vrednošću z prouzrokuje da perspektivna projekcija udaljenijih objekata bude manja od projekcije bližih objekata. Pritom vrednosti ovog preslikavanja nisu definisane za $z = 0$.

4.3.3 Kosa projekcija

Razmotrimo sada primer kose projekcije na ravan $z = 0$ kod koje je ugao koji pravac projektovanja zahvata sa ravni projektovanja jednak φ. Tačka $P(x, y, z)$ projektuje se u tačku $P_p(x_p, y_p, 0)$ (slika 4.27). Ortogonalna projekcija Q tačke P na ravan $z = 0$ imaće koordinate $Q(x, y, 0)$. Neka duž P_pQ zahvata ugao α sa x osom (to je ugao α sa slika 4.21 i 4.22 koji određuje veličinu ugla koji projekcije duži upravne na ravan projekcije zahvataju sa x osom) i označimo sa L dužinu duži P_pQ.



Slika 4.27: Koso projektovanje na ravan $z = 0$.

Trougao PP_pQ je pravougli sa pravim uglom kod temena Q i iz njega sledi:

$$\tan\varphi = \frac{|PQ|}{|P_pQ|} = \frac{z}{L}$$

te L možemo izraziti kao:

$$L = \frac{z}{\tan\varphi} = z \cdot \cot\varphi \tag{4.1}$$

Ako iz tačke P_p spustimo normalu na pravu iz tačke Q paralelnu x osi i označimo presečnu tačku sa R, dobijamo pravougli trougao P_pRQ sa uglom α iz koga sledi:

$$\begin{aligned} \sin\alpha &= \frac{y_p - y}{L} \\ \cos\alpha &= \frac{x_p - x}{L} \end{aligned}$$

Dakle, koordinate tačke P_p mogu se izraziti kao:

$$\begin{aligned} x_p &= x + L \cdot \cos\alpha \\ y_p &= y + L \cdot \sin\alpha \end{aligned} \tag{4.2}$$

a odavde, zamenom vrednosti L iz jednačine (4.1) u jednačine (4.2), dobijamo jednačine kosog projektovanja na ravan $z = 0$:

$$\begin{aligned} x_p &= x + z \cdot \cot\varphi \cdot \cos\alpha \\ y_p &= y + z \cdot \cot\varphi \cdot \sin\alpha \\ z_p &= 0 \end{aligned}$$

Matrica kosog projektovanja jednaka je:

$$M_{koso} = \begin{bmatrix} 1 & 0 & \cot\varphi \cdot \cos\alpha & 0 \\ 0 & 1 & \cot\varphi \cdot \sin\alpha & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Specijalno, za cavalier projektovanje važi $\varphi = 45°$ i $\cot\varphi = 1$ te je matrica cavalier projektovanja jednaka:

$$M_{cav} = \begin{bmatrix} 1 & 0 & \cos\alpha & 0 \\ 0 & 1 & \sin\alpha & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

dok za cabinet projektovanje važi $\varphi = \arctan 2$ i $\cot\varphi = 1/\tan\varphi = 1/\tan(\arctan 2) = 1/2$ te je matrica cabinet projektovanja jednaka:

$$M_{cab} = \begin{bmatrix} 1 & 0 & \frac{1}{2}\cos\alpha & 0 \\ 0 & 1 & \frac{1}{2}\sin\alpha & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

U slučaju ortogonalne projekcije, važi $\varphi = \pi/2$ i $\cot\varphi = 0$ i dobijaju se jednačine:

$$x_p = x, \quad y_p = y, \quad z_p = 0$$

Primetimo da se matrica kosog projektovanja može izraziti kao kompozicija smicanja u pravcu Oxy koordinatne ravni i ortografskog projektovanja na ravan $z = 0$:

$$M_{koso} = M_{orth} \cdot SH_z$$

.

4.4 Pitanja

4.1 Šta je projekcija? Šta znači da je projekcija planarna?

4.2 Čime je određena projekcija?

4.3 Kako se mogu klasifikovati projekcije iz trodimenzionog prostora u ravan? U odnosu na šta se vrši klasifikacija?

4.4 Koje su karakteristike perspektivne projekcije?

4.5 Da li se paralelne prave slikaju u paralelne pri perspektivnoj projekciji? Odgovor obrazložiti.

4.6 Kako nazivamo tačku u kojoj se seku paralelne prave kod perspektivne projekcije? Koliko može biti takvih tačaka?

4.7 Kako nazivamo tačku u kojoj se seku prave paralelne nekoj koordinatnoj osi kod perspektivne projekcije? Koliko najviše takvih tačaka može postojati?

4.8 Kako se dele ortogonalne projekcije u odnosu na ugao koji zahvata ravan projekcije sa koordinatnim osama?

4.9 Koliko tipova ortografskih projekcija razlikujemo?

4.10 Prema čemu se dele aksonometrijske projekcije?

4.11 Opisati Bruneleskijev eksperiment.

4.12 Kakvo je skraćenje kod aksonometrijskog projektovanja, a kakvo kod perspektivnog projektovanja?

4.13 Šta važi za vektor normale ravni projekcije kod izometrijskog projektovanja?

4.14 Šta važi za ravan projekcije, a šta za pravac projektovanja kod kosog projektovanja?

4.15 Nabrojati prednosti i mane kosog projektovanja.

4.16 Koja su dva najčešća tipa kosih projekcija? Čemu je jednak ugao između pravca projektovanja i normale ravni projektovanja kod svakog od njih?

4.17 Koliku dužinu imaju projekcije duži paralelne sa ravni projekcije kod cavalier projektovanja, a koliku kod cabinet projektovanja?

4.18 Koliku dužinu imaju projekcije duži upravnih na ravan projekcije kod cavalier projektovanja, a koliku kod cabinet projektovanja?

4.19 Prema čemu se međusobno mogu razlikovati cabinet projekcije?

4.20 Kod kojih od navedenih tipova projekcija je ravan projekcije upravna na koordinatnu osu: (a) ortografska projekcija (b) izometrijska projekcija (c) cavalier projekcija?

4.21 Kod kojih od navedenih tipova projekcija je projekciona ravan upravna na pravac projekcije: (a) aksonometrijska projekcija (b) ortografska projekcija (c) cabinet projekcija?

4.22 Kod kojih od navedenih tipova projekcija normala ravni projekcije zahvata jednak ugao sa sve tri koordinatne ose (a) kod aksonometrijske (b) kod izometrijske (c) kod trimetrijske projekcije?

4.23 Napisati matricu transformacije u homogenim koordinatama koja odgovara ortografskoj projekciji na ravan $x = 0$.

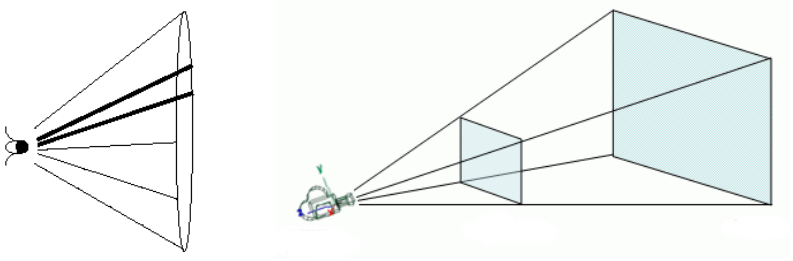
4.24 Napisati matricu transformacije u homogenim koordinatama koja odgovara perspektivnom preslikavanju kod koga je ravan projektovanja ravan $z = 0$, a centar projekcije sa koordinatama $(0, 0, -d)$.

Glava 5

Model sintetičke kamere

Zašto ne renderovati sve što možemo da vidimo? Čovekovo vidno polje iznosi oko $180°$. Međutim, čak i ako bismo iz vidnog polja odbacili deo koji možemo detektovati perifernim vidom i ograničili se na centralni deo vidnog polja koji iznosi oko $120°$, slike koje bi obuhvatale ovako široko vidno polje delovale bi neprirodno i iskrivljeno. Razlog za to je taj što čovek uobičajeno posmatra scene koje zauzimaju relativno mali deo našeg vidnog polja. Na primer, računarski monitor na ugodnoj vidnoj distanci zauzima samo $25°$ od ukupnog vidnog polja, dok ekran telefona zauzima samo nekoliko stepeni vidnog polja. Stoga je bolje ne renderovati celokupno vidno polje, već, kao i u pristupu koji koriste fotografi, renderovati samo skroman deo vidnog polja.

Zapremina pogleda (eng. view volume) sadrži sve ono što kamera vidi. Objekti koji se nalaze unutar zapremine pogleda biće prikazani na slici, objekti koji su u potpunosti van nje se odbacuju, dok se kod objekata čiji se jedan deo nalazi van zapremine pogleda, deo van zapremine pogleda odbacuje. Zapreminu pogleda modelujemo kao konačni (zatvoreni) opseg trodimenzionog prostora i U slučaju oka zapremina pogleda ima oblik kupe, međutim, pošto je matematika potrebna za odsecanje objekata u odnosu na konusnu površ skupa, aproksimiramo je četvorostranom piramidom (slika 5.1).



Slika 5.1: a) Konusna perspektivna zapremina pogleda b) zapremina pogleda u obliku zarubljene četvorostrane piramide (preuzeto sa U Brown).

Fotoaparat na ulazu prima trodimenzionu scenu i transformiše je u njenu dvodimenzionu reprezentaciju koja se može smestiti na filmsku rolnu (u slučaju analogne fotografije) ili biti zadata u vidu niza piksela (kod digitalne fotografije). Kako bi se putem fotoaparata napravila odgovarajuća slika potrebno je pažljivo zadati vrednosti različitih parametara: to su pre svega pozicija fotoaparata, njegova orijentacija i vidno polje (koje se podešava kontrolom zuma na fotoaparatu). Za bolje fotoaparate, potrebno je postaviti i vrednost žižne daljine (rastojanja tačaka koje su u najvećem fokusu na slici) i dubine polja (koliko daleko ispred i iza žižne daljine će objekti biti u fokusu). Takođe, nekad je moguće zadati i nagib ravni slike: ukoliko ravan slike nije upravna na pravac pogleda dobija se kosa projekcija. U nastavku teksta termine fotoaparat i kamera koristićemo kao sinonime.

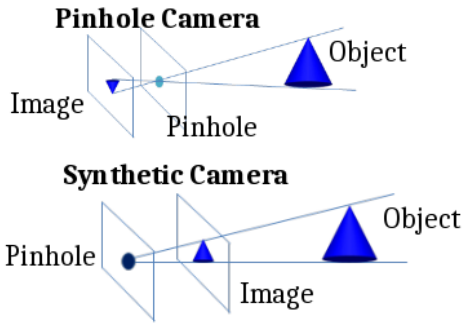
Kako bismo u računarskoj grafici modelovali ponašanje kamere/fotoaparata, koristimo *model sintetičke kamere* (eng. synthetic camera model). Razmatraćemo dva modela sintetičke kamere: jedan u slučaju perspektivnog projektovanja (kada je kamera na konačnom rastojanju od objekata) i drugi u slučaju paralelnog projektovanja (kada je kamera na beskonačno dalekom rastojanju od objekata).

5.1 Zadavanje kamere kod perspektivnog projektovanja

Sintetička kamera kod perspektivnog projektovanja u velikoj meri nalikuje tzv. *tačkastoj kameri* (eng. pinhole camera), koja prati koncept kamere opskura (lat. camera obscura). Kamera opskura predstavlja optičku spravu, preteču fotoaparata, koja se u prošlosti, još od šesnaestog veka, koristila za dobijanje realističnih slika. Ona funkcioniše po sledećem principu: imamo zamračenu sobu ili kutiju koja ima mali otvor ili sočivo kroz koje zraci svetlosti prolaze i padaju na zadnji zid aparature na kome se projektuje sadržaj scene (slika 5.3 a)). Jasno je da ovaj princip dobijanja slike odgovara perspektivnoj projekciji, pri čemu otvor kroz koji prolaze zraci odgovara centru projekcije. Ravan na koju se slika projektuje se nalazi sa druge strane centra projekcije u odnosu na scenu što rezultuje time da je slika scene prikazana naopako. Kameru opskuru su koristili mnogi hiperrealisti poput Karavađa, Velaskeza i Da Vinčija za kreiranje svojih remek-dela.

Slika 5.2: Ilustracija dobijanja slike korišćenjem kamere opskure (preuzeto sa http://thedelightsofseeing.blogspot.com/2010/10/pinhole-photography-and-camera-obscura.html).

Kod sintetičke kamere zraci takođe ulaze u kameru kroz jedinstvenu tačku koja predstavlja centar projekcije, međutim, za razliku od tačkaste kamere, kod sintetičke kamere se centar projekcije nalazi iza ravni projektovanja u odnosu na scenu koja se projektuje (slika 5.3).



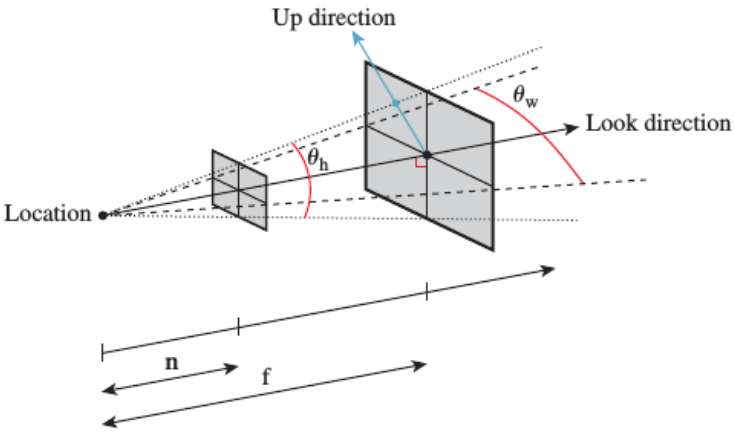
Slika 5.3: Prikaz funkcionisanja tačkaste i sintetičke kamere.

Zapremina pogleda je u modelu sintetičke kamere kod perspektivnog projektovanja, u potpunosti određena vrednostima narednih šest parametara:

- pozicije kamere,
- vektora pogleda i vektora nagore,
- vidnog polja i
- prednje i zadnje ravni odsecanja.

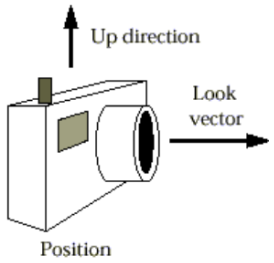
U modelu sintetičke kamere najčešće se ne razmatraju parametri poput žižne daljine i dubine polja jer je idealna kamera u fokusu na svim rastojanjima. Na slici 5.4 ilustrovani su osnovni parametri sintetičke kamere.

Pozicija kamere predstavlja centar projekcije i, kao što smo već pomenuli, ona se u modelu sintetičke kamere nalazi iza ravni projektovanja u odnosu na objekat koji se projektuje. Zadaje se tačkom u svetskom koordinatnom sistemu.



Slika 5.4: Specifikacija sintetičke kamere (preuzeto iz knjige CG: Principles and Practice).

Smer i orijentacija kamere zadaju se korišćenjem *vektora pogleda* (eng. look vector) i *vektora nagore* (eng. up vector) (slika 5.5). Vektorom pogleda se zadaje smer u kome je kamera usmerena. Ukoliko bismo pustili zrak iz pozicije kamere usmeren u pravcu vektora pogleda „udarili" bismo u objekat koji će biti prikazan u središtu slike dobijene sintetičkom kamerom. U praksi, smer pogleda kamere se obično ne zadaje direktno, već se pored pozicije kamere zadaje i tačka u koju kamera gleda i na osnovu te dve tačke se računa vektor pogleda.

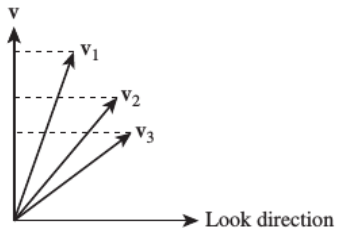


Slika 5.5: Pozicija, vektor pogleda i vektor nagore (preuzeto iz knjige CG: Principles and Practice).

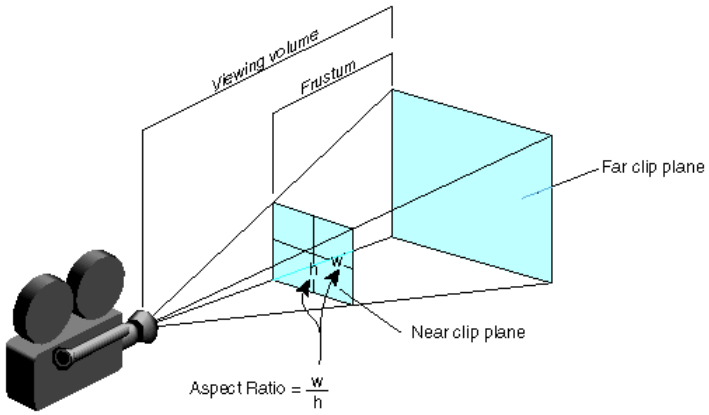
Potrebno je zadati i vertikalno usmerenje kamere, tj. orijentisati kameru oko vektora pogleda: na primer, da li držimo kameru u horizontalnom ili vertikalnom položaju ili u nekom međupoložaju (slika 5.5). Ako zamislimo vertikalni jedinični vektor v nacrtan na pozadini kamere, on bi bio upravan na vektor pogleda i zajedno sa vektorom pogleda određivao bi ravan koja se naziva vertikalna ravan kamere. Pokazuje se da u praksi nije tako jednostavno zadati vektor ortogonalan na vektor pogleda, te se od korisnika zahteva da kao vektor nagore zada proizvoljan nenula vektor u vertikalnoj ravni kamere (koji je različit od vektora pogleda) i na osnovu njega se jednostavno računa vektor v. Na slici 5.6 ilustrovan je ovaj princip: bilo koji od vektora v_1, v_2 i v_3 može od strane korisnika biti zadat kao vektor nagore[1], a vektor v se zatim računa kao vektor u vertikalnoj ravni kamere upravan na vektor pogleda. Dakle, *orijentacija kamere* definisana je jediničnim vektorom v upravnim na vektor pogleda u ravni definisanoj vektorom pogleda i vektorom nagore (tj. u vertikalnoj ravni kamere).

Vidno polje kamere se zadaje *uglovima vidnog polja*. Oni definišu koliko široko (u horizontalnom i vertikalnom smeru), izraženo u stepenima, kamera može da „vidi" (uglovi Θ_w i Θ_h na slici 5.4). Najjednostavniji model sintetičke kamere proizvodi kvadratnu sliku pa je vidno polje isto i u horizontalnom i u vertikalnom smeru. Ipak, u opštem slučaju proizvedena slika je pravougaonog oblika i potrebno je zadati posebno horizontalni, a posebno vertikalni ugao vidnog polja. U nekim sistemima uobičajeno je zadati ugao horizontalnog vidnog polja i *odnos širine i visine* (eng. aspect ratio) prozora za prikaz, dok se ugao vertikalnog vidnog polja računa na osnovu vrednosti tih podataka (slika 5.7). Odnos širine i visine definiše dimenziju platna na koju se scena projektuje, nakon čega se ona preslikava u oblast za prikaz; dobra praksa je da ovaj odnos bude isti za prozor pogleda i oblast za prikaz da ne bi došlo do deformisanja (istezanja ili suženja) slike.

[1]Obratimo pažnju na to da termin vektor nagore može navoditi na zaključak da je on upravan na vektor pogleda i da zadaje orijentaciju kamere, međutim to ne mora biti slučaj.

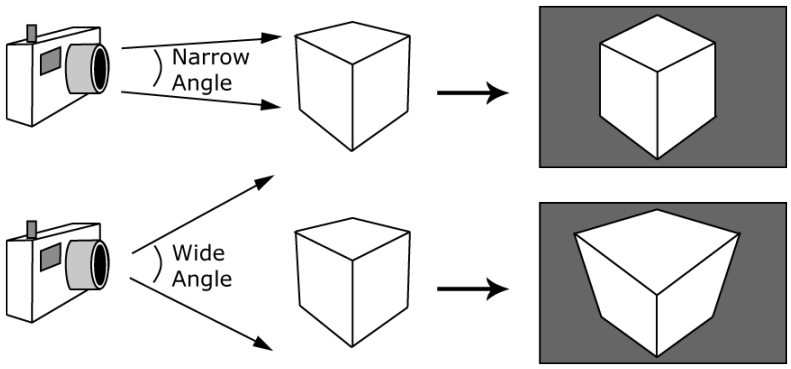


Slika 5.6: Bilo koji od vektora v_1, v_2, v_3 može biti zadat kao vektor nagore, a onda se vektor v računa kao vektor u ravni određenoj vektorom pogleda i vektorom nagore, koji je upravan na vektor pogleda (preuzeto iz knjige: CG: Principles and Practice).



Slika 5.7: Zadavanje vidnog polja preko odnosa širine (w) i visine (h) prozora za prikaz (preuzeto sa U Brown).

Vidno polje određuje koji deo scene će stati u zapreminu pogleda – što je veći ugao vidnog polja to će veći deo scene stati u zapreminu pogleda. Na ovaj način se, međutim, zadaje i količina perspektivnog iskrivljenja na slici (slika 5.8), čije su moguće vrednosti od gotovo nula (kod teleobjektiva, namenjenih za slikanje udaljenih objekata kod kojih je projekcija skoro paralelna) do velikih vrednosti perspektivnog iskrivljenja (kod širokougaonih objektiva, namenjenih fotografisanju većih grupa ljudi ili širokih prostranstava poput trgova). Sa povećanjem žižne daljine povećava se motiv u kadru, ali se istovremeno smanjuje vidni ugao[2].

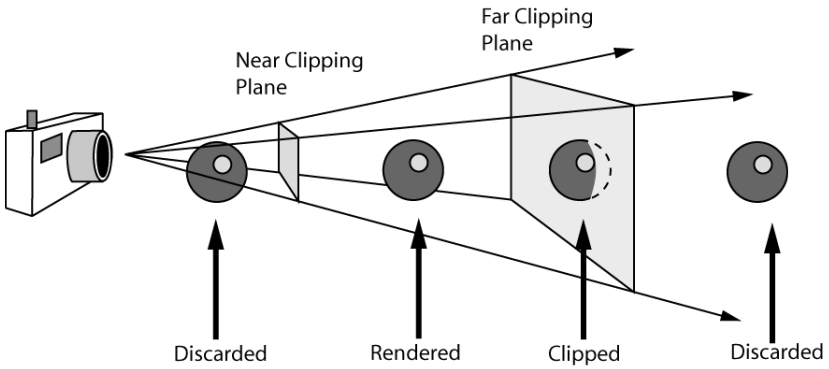


Slika 5.8: Slika dobijena korišćenjem sočiva sa uskim i širokim vidnim uglom (preuzeto sa U Brown).

Do sada razmatranim parametrima indirektno smo opisali četvorostranu zapreminu pogleda sa pravougaonim poprečnim presekom. Preostalo je da se zadaju još dve komponente: prednja i zadnja ravan odsecanja. *Prednja i zadnja ravan odsecanja* (eng. near and far clipping plane) su paralelne ravni projektovanja i zadaju se svojim rastojanjima od kamere. Uobičajeno se sa n (od eng. near) označava rastojanje prednje ravni odsecanja od kamere, a sa f (od eng. far) rastojanje zadnje ravni odsecanja

[2]Zumiranjem je moguće promeniti žižnu daljinu objektiva.

od kamere (slika 5.4). Prednja i zadnja ravan odsecanja zajedno sa prethodno navedenim parametrima formiraju *zarubljenu četvorostranu piramidu pogleda* (eng. frustum). Na ovaj način ograničava se domet pogleda kamere renderovanjem samo objekata, odnosno delova objekata, koji se nalaze između prednje i zadnje ravni odsecanja i odsecanjem svega van njih (slika 5.9). Ovo predstavlja koristan mehanizam jer često ne želimo da prikazujemo objekte koji su preblizu kameri (jer bi blokirali ostatak scene i imali visok stepen iskrivljenja) ili predaleko od kamere (jer su isuviše mali da bi bili vizualno značajni, a mogu uzeti značajno vreme za renderovanje). Na ovaj način eliminiše se i razmatranje objekata koji su iza kamere. Sve ovo donosi uštede u ukupnom vremenu renderovanja.



Slika 5.9: Primer odsecanja objekata u odnosu na prednju i zadnju ravan odsecanja (preuzeto sa U Brown).

Koncept prednje i zadnje ravni odsecanja je često korišćen u industriji računarskih igara kako bi se eliminisali udaljeni objekti. On može proizvesti efekat da se, dok se tokom igre krećemo unapred, udaljeni objekat odjednom pojavi na ekranu. Jedna od opštih tehnika za rešavanje ovog nepoželjnog efekta koja se dugo koristila u industriji igara jeste da se objekti u daljini renderuju korišćenjem pozadinske magle te da se oni postepeno pojavljuju prilikom prilaska objektu. U novijim igrama koriste se bolji sistemi renderovanja koji zadnju ravan odsecanja postavljaju dovoljno daleko, dok objekte prikazuju sa različitim nivoom detaljnosti: udaljeniji objekti iscrtavaju se korišćenjem manjeg broja poligona, čime se povećava efikasnost renderovanja.

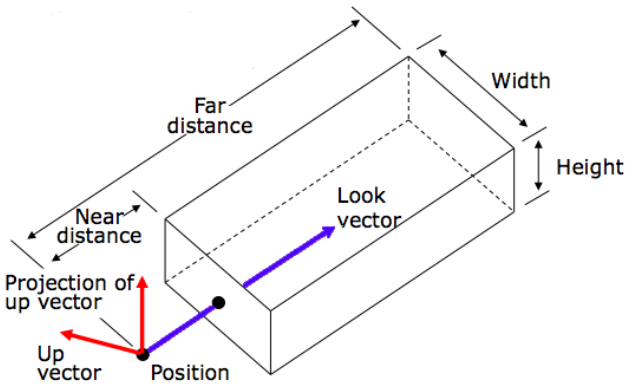
5.2 Zadavanje kamere kod paralelnog projektovanja

Zapremina pogleda se u slučaju paralelnog projektovanja zadaje istim skupom parametara kao i pri perspektivnom projektovanju, osim uglova vidnog polja. Naime, pošto je paralelna zapremina pogleda u obliku paralelopipeda, umesto uglova kojima se karakteriše vidno polje, potrebno je zadati visinu i širinu zapremine pogleda (slika 5.10). U slučaju ortogonalnog projektovanja zapremina pogleda biće u obliku pravougaonog paralelopipeda, dok kod kosog projektovanja to neće biti slučaj. Veličina prikazanog objekta pri perspektivnom projektovanju zavisi udaljenosti objekta od kamere, dok se kod paralelnog projektovanja objekti prikazuju u istoj veličini koliko god bili udaljeni od kamere jer su svi projektivni zraci međusobno paralelni. Skraćenje je pri paralelnom projektovanju uniformno i jedino zavisi od ugla koji zraci projekcije zahvataju sa ravni projekcije; dakle, kao što smo i ranije istakli, ne postoji perspektiva zavisna od dubine. Iz tog razloga se u slučaju paralelne zapremine pogleda projektovanje vrši jednostavnije, a takođe i odsecanje jer ravni u odnosu na koje vršimo odsecanje imaju jednostavnije jednačine.

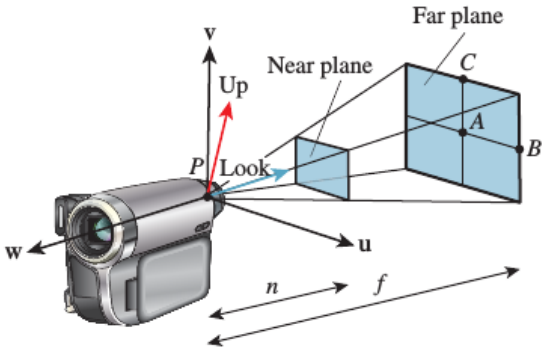
5.3 Izgradnja matrica transformacija na osnovu zadatih parametara kamere

Razmotrimo sada kako da na osnovu zadatih parametara sintetičke kamere kod perspektivnog projektovanja izvedemo:

- jednačine ortonormiranog koordinatnog sistema (u, v, w) sa koordinatnim početkom u središtu sočiva kamere – nadalje ćemo ga zvati *koordinatni sistem kamere*,
- koordinate nekoliko tačaka na zarubljenoj piramidi pogleda (tačaka A, B i C na slici 5.11).



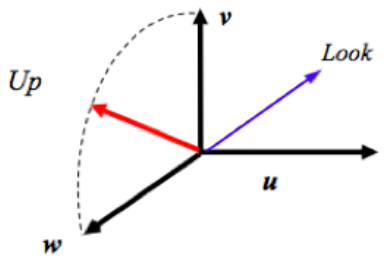
Slika 5.10: Paralelna zapremina pogleda (preuzeto sa U Brown).



Slika 5.11: Koordinatni sistem kamere (u, v, w), vektor pogleda $look$, vektor nagore up i vektor v, tačke P, A, B i C (preuzeto iz knjige CG: Principles and Practice).

Označimo poziciju sintetičke kamere sa P, vektor pogleda sa $\overrightarrow{look}$, vektor nagore sa $\overrightarrow{up}$, uglove horizontalnog i vertikalnog vidnog polja izražene u stepenima sa Θ_h^{deg} i Θ_v^{deg}, a rastojanja prednje i zadnje ravni odsecanja od kamere sa n i f, redom.

Razmotrimo, najpre, kako izgraditi koordinatni sistem kamere (u, v, w). Pritom želimo da prva koordinatna osa (u ovom slučaju osa u) bude usmerena nadesno u odnosu na kameru, druga koordinatna osa (osa v) bude usmerena naviše u odnosu na kameru, a treća koordinatna osa (osa w) usmerena tako da kamera gleda duž negativnog smera w ose. To ćemo postići tako što će vektor $\overrightarrow{look}$ ležati duž negativne ose w, osu v ćemo definisati kao vektor upravan na vektor $\overrightarrow{look}$ koji leži u ravni određenoj vektorima $\overrightarrow{look}$ i $\overrightarrow{up}$, dok će osa u biti upravna na v i w osu i sa njima će formirati pozitivno orijentisani koordinatni sistem (slika 5.12).



Slika 5.12: Koordinatni sistem kamere.

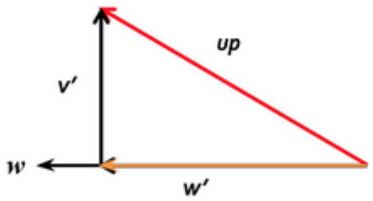
Gradimo koordinatni sistem (u, v, w) u obrnutom redosledu koordinatnih osa. Vektor $\overrightarrow{w}$ računamo

kao jedinični vektor suprotnog smera od smera vektora pogleda:

$$\overrightarrow{w} = \frac{-\overrightarrow{look}}{||\overrightarrow{look}||}$$

Kako bismo konstruisali vektor $\overrightarrow{v}$, najpre ćemo odrediti projekciju vektora $\overrightarrow{up}$ na ravan upravnu na vektor $\overrightarrow{w}$ i taj vektor označiti sa $\overrightarrow{v'}$, a zatim odrediti jedinični vektor u tom smeru. Vektor $\overrightarrow{v'}$ možemo izračunati oduzimanjem od vektora $\overrightarrow{up}$ komponente vektora $\overrightarrow{up}$ u pravcu vektora $\overrightarrow{w}$ (koju ćemo označiti sa $\overrightarrow{w'}$) (slika 5.13):

$$\overrightarrow{v'} = \overrightarrow{up} - proj_{\overrightarrow{w}}\overrightarrow{up} = \overrightarrow{up} - \overrightarrow{w'}$$



Slika 5.13: Računanje vektora v' kao vektora dobijenog oduzimanjem od vektora up komponente u pravcu vektora w (preuzeto iz slajdova: U Brown).

S obzirom na to da je vektor $\overrightarrow{w}$ intenziteta 1, ako sa φ označimo ugao koji zaklapaju vektor $\overrightarrow{up}$ i vektor $\overrightarrow{w'}$ važi:

$$|\overrightarrow{w'}| = |\overrightarrow{up}|\cos\varphi = |\overrightarrow{up}|\cos\varphi|\overrightarrow{w}| = \overrightarrow{up}\cdot\overrightarrow{w}$$

te je:

$$\overrightarrow{w'} = (\overrightarrow{up}\cdot\overrightarrow{w})\overrightarrow{w}$$

odnosno važi:

$$\overrightarrow{v'} = \overrightarrow{up} - (\overrightarrow{up}\cdot\overrightarrow{w})\overrightarrow{w}$$

odakle normiranjem vektora $\overrightarrow{v'}$ dobijamo vektor $\overrightarrow{v}$:

$$\overrightarrow{v} = \frac{\overrightarrow{v'}}{||\overrightarrow{v'}||}$$

Konačno, da bismo dobili pozitivo orijentisani ortonormirani sistem, vektor $\overrightarrow{u}$ se računa prema formuli:

$$\overrightarrow{u} = \overrightarrow{v}\times\overrightarrow{w}$$

Izračunajmo sada koordinate tačaka A, B i C sa slike 5.11. Tačka A se nalazi u smeru vektora $-\overrightarrow{w}$ na rastojanju f od tačke P te važi:

$$\overrightarrow{PA} = -f\cdot\overrightarrow{w}$$

Ivica AB je naspram polovine horizontalnog vidnog polja iz P i na rastojanju je f od P, te iz $\triangle PAB$ sledi:

$$|AB| = f\tan\frac{\Theta_h}{2}$$

gde je sa Θ_h označen horizontalni vidni ugao, konvertovan iz stepena u radijane, odnosno:

$$\Theta_h = \Theta_h^{deg}\frac{\pi}{180}$$

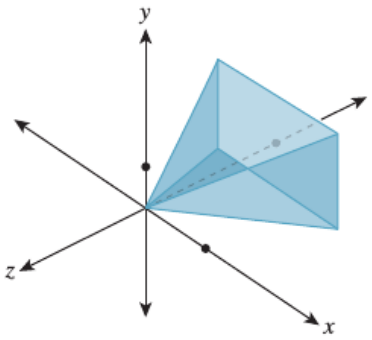
Na sličan način se računa i vertikalno vidno polje Θ_v i dužina duži AC, odnosno važi:

$$A = P - f\cdot\overrightarrow{w}$$
$$B = A + f\tan\tfrac{\Theta_h}{2}\overrightarrow{u} = P + f\tan\tfrac{\Theta_h}{2}\overrightarrow{u} - f\cdot\overrightarrow{w}$$
$$C = A + f\tan\tfrac{\Theta_v}{2}\overrightarrow{v} = P + f\tan\tfrac{\Theta_v}{2}\overrightarrow{v} - f\cdot\overrightarrow{w}$$

S obzirom na to da smo jedinične vektore koordinatnog sistema kamere izrazili preko parametara sintetičke kamere, a koordinate tačaka A, B i C u koordinatnom sistemu kamere, možemo zaključiti da smo koordinata tačaka P, A, B i C izrazili preko narednih parametara sintetičke kamere: pozicije kamere, vektora pogleda i nagore, uglova vidnog polja i rastojanja od zadnje ravni odsecanja. Može se primetiti da nigde u dosadašnjim izvođenjima nije korišćeno rastojanje od prednje ravni odsecanja n.

5.3.1 Standardna perspektivna zapremina pogleda

Iskoristićemo tačke P, A, B i C za transformisanje zarubljene piramide pogleda date sintetičke kamere u kanonsku (standardnu) zarubljenu piramidu pogleda prikazanu na slici 5.14. *Standardna perspektivna zapremina pogleda* (eng. standard perspective view volume) je četvorostrana piramida čiji se vrh nalazi u koordinatnom početku i čije su x i y koordinate u opsegu od -1 do 1, a z koordinata u opsegu od 0 do -1 (slika 5.14).



Slika 5.14: Standardna perspektivna zapremina pogleda (skala po z je namerno preuveličana na slici).

Transformacija koja preslikava proizvoljnu perpsektivnu zapreminu pogleda u standardnu perspektivnu zapreminu pogleda je afina transformacija u trodimenzionom prostoru i kao takva određena je slikama četiri nekomplanarne tačke. Dakle, sve što treba uraditi kako bismo zadali ovu transformaciju jeste zadati gde tačke P, A, B i C treba da se preslikaju. Tačku P je potrebno preslikati u koordinatni početak, tačku A u središte osnove piramide koje ima koordinate $(0, 0, -1)$, tačku B u središte desne ivice osnove piramide koje ima koordinate $(1, 0, -1)$, a tačku C u središte gornje ivice osnove koje ima koordinate $(0, 1, -1)$. Ovom transformacijom se tačke zadnje ravni odsecanja datog modela sintetičke kamere transformišu u ravan $z = -1$. S obzirom na to da se rastojanja duž zraka iz tačke P do tačke A transformišu linearno (u pitanju je skaliranje sa centrom u koordinatnom početku), tačke sa prednje ravni odsecanja se transformišu u ravan $z = -n/f$ (ovo se može dobiti iz odnosa $n : f = z : -1$ jer se zadnja ravan odsecanja slika u ravan $z = -1$). Svođenjem proizvoljne zapremine pogleda na kanonsku postiže se da se sve dalje operacije izvode nezavisno od vrednosti parametara sintetičke kamere (osim što će faktor $-n/f$ ući u neka računanja). Dakle, kod standardne perspektivne zapremine pogleda vrednosti svih parametara sintetičke kamere su fiksirane, osim jednačine prednje ravni odsecanja koja zavisi od parametara n i f polazne perspektivne zapremine pogleda.

Matricu M_{persp} transformacije proizvoljne u standardnu perspektivnu zapreminu pogleda možemo izvesti u nekoliko koraka:

- koristimo poziciju kamere P, vektore $\overrightarrow{look}$ i $\overrightarrow{up}$ da odredimo koordinatni sistem kamere (u, v, w),
- primenjujemo translaciju kojom tačku P slikamo u koordinatni početak,
- primenjujemo odgovarajući broj rotacija oko koordinatnih osa tako da se koordinatne ose koordinatnog sistema kamere (u, v, w) dovedu na poklapanje sa osama koordinatnog sistema (x, y, z),
- primenjujemo skaliranje sa koeficijentom $1/f$ po z osi tako da se zadnja ravan odsecanja sa jednačinom $z = -f$ preslika u ravan $z = -1$,
- izvodimo skaliranje sa koeficijentima $1/(f \tan \frac{\Theta_h}{2})$ i $1/(f \tan \frac{\Theta_v}{2})$ po x i y osi tako da visina i širina zarubljene piramide pogleda budu jednake 2.

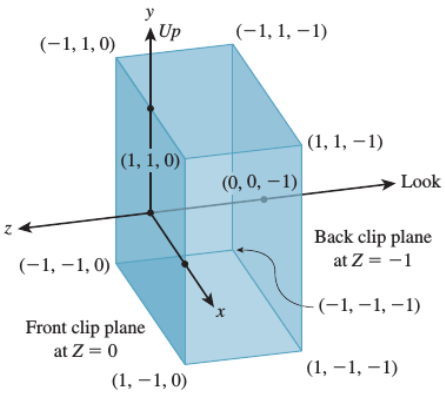
Ako tačka P i vektori $\overrightarrow{u}$, $\overrightarrow{v}$ i $\overrightarrow{w}$ u koordinatnom sistemu (x, y, z) imaju redom koordinate $P(x_P, y_P, z_P)$, $\overrightarrow{u}(u_x, u_y, u_z)$, $\overrightarrow{v}(v_x, v_y, v_z)$, $\overrightarrow{w}(w_x, w_y, w_z)$ matrica transformacije biće jednaka:

$$M_{persp} = \begin{bmatrix} \frac{1}{f \tan \frac{\Theta_h}{2}} & 0 & 0 & 0 \\ 0 & \frac{1}{f \tan \frac{\Theta_v}{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{f} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -x_p \\ 0 & 1 & 0 & -y_P \\ 0 & 0 & 1 & -z_P \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Desna matrica u ovom proizvodu odgovara translaciji tačke P u koordinatni početak, srednja matrica transformiše vektor $\overrightarrow{u}$ u $\overrightarrow{e_1}$, $\overrightarrow{v}$ u $\overrightarrow{e_2}$ i $\overrightarrow{w}$ u $\overrightarrow{e_3}$, a leva matrica skalira svaku od koordinatnih osa na odgovarajući način. Zanimljivo je primetiti da matrica transformacije koja slika vektor $\overrightarrow{e_1}$ u $\overrightarrow{u}$, vektor $\overrightarrow{e_2}$ u $\overrightarrow{v}$ i $\overrightarrow{e_3}$ u $\overrightarrow{w}$ ima po kolonama redom koordinate vektora $\overrightarrow{u}$, $\overrightarrow{v}$ i $\overrightarrow{w}$ i predstavlja rotaciju. Pošto je ovde potrebno primeniti njoj inverznu transformaciju, a inverz rotacije je opet rotacija sa matricom transformacije koja je njoj transponovana, potrebnu matricu dobijamo upisivanjem koordinata vektora $\overrightarrow{u}$, $\overrightarrow{v}$ i $\overrightarrow{w}$ po vrstama.

Tačke standardne perspektivne zapremine pogleda je moguće jednostavno projektovati na ravan paralelnu ravni $z = 0$, na primer na ravan $z = 1$, korišćenjem perspektivne projekcije sa centrom projekcije u koordinatnom početku: $(x, y, z) \to (x/z, y/z, 1)$. Međutim, umesto da na ovaj način računamo projekciju, vodimo se drugom idejom: nakon transformacije proizvoljne perspektivne zapremine pogleda u kanonsku zapreminu pogleda primenićemo još dve transformacije: prvu, kojom ćemo razviti piramidalnu zapreminu pogleda u pravougaoni paralelopiped, i drugu, kojom ćemo izvršiti projektovanje duž z ose. Na ovaj način se izračunavanja koja se vrše u grafičkoj jedinici pojednostavljuju i čine efikasnijim. Na primer, prilikom projektovanja duž z ose (kod paralelnog projektovanja), veoma je jednostavno odrediti koji objekti zaklanjaju neke druge objekte, odnosno testiranje vidljivosti postaje trivijalno (što će biti od koristi u dizajniranju z-bafer algoritma). Na ovaj način umesto da radimo sa matricama koje odgovaraju različitim vrstama projektovanja, sve svodimo na ortografsku projekciju sa standardnom zapreminom pogleda.

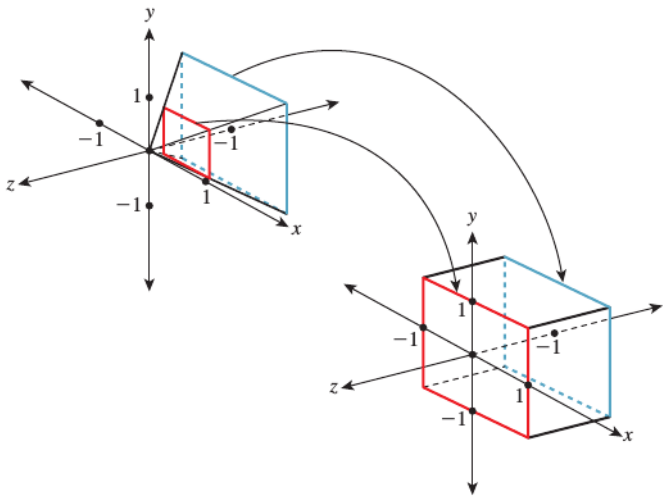
5.3.2 Standardna paralelna zapremina pogleda



Slika 5.15: Standardna paralelna zapremina pogleda.

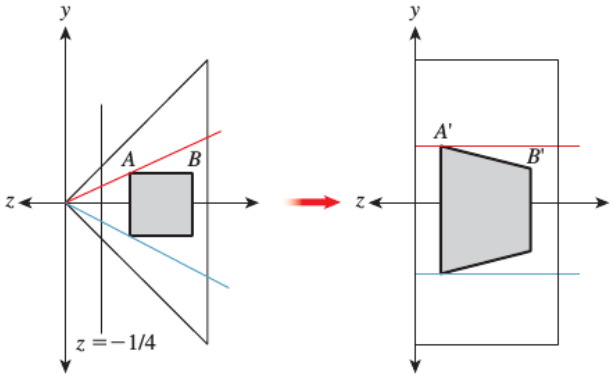
Standardna paralelna zapremina pogleda (eng. standard parallel view volume) je pravougaoni paralelopiped čije granice idu od -1 do 1 po x i y osi i od 0 do -1 po z osi (slika 5.15). Njena prednja ravan odsecanja ima jednačinu $z = 0$, a zadnja $z = -1$. Koordinatni početak se nalazi u centru prednje ravni odsecanja. Razmotrimo transformaciju kojom se deo standardne perspektivne zapremine pogleda između prednje i zadnje ravni odsecanja (deo između ravni $z = -n/f$ i ravni $z = -1$) transformiše u standardnu paralelnu zapreminu pogleda. Ova transformacija preslikava kvadrat prednje ravni odsecanja perspektivne zapremine pogleda u odgovarajući kvadrat prednje ravni odsecanja paralelne zapremine pogleda (slika 5.16). Ovom transformacijom se prave slikaju u prave, specijalno zraci kroz koordinatni početak koji prolaze kroz standardnu perspektivnu zapreminu pogleda transformišu se u zrake paralelne z osi koji prolaze kroz standardnu paralelnu zapreminu pogleda.

Primena ove transformacije ne utiče na finalni rezultat procesa renderovanja jer perspektivno projektovanje koje je oblika $(x, y, z) \to (x/z, y/z, 1)$ u zapremini pogleda pre transformacije odgovara paralelnom projektovanju $(x, y, z) \to (x, y, 1)$ transformisanog oblika u zapremini pogleda nakon ove transformacije. Ovo je jednostavno uočiti razmatranjem projekcije na Oyz ravan. Pogledajmo npr. kvadrat prikazan na slici 5.17 koji zauzima srednju polovinu perspektivnog pogleda na scenu. Informacija o tome koje tačke su zaklonjene nekim drugima se utvrđuje poretkom tačaka duž zraka od tačke pogleda do scene, te je tačka B zaklonjena bližom ivicom kvadrata. Nakon transformacije ovaj zrak postaje zrak u smeru $-z$ i možemo primetiti da je tačka B' takođe zaklonjena prednjom ivicom trapeza dobijenog kao slika kvadrata. Stranica kvadrata sa slike 5.17 koja je bliža prednjoj ravni odsecanja nakon primene transformacije biće prikazana kao duža od slike udaljenije stranice čime se postiže efekat perspektivnog



Slika 5.16: Transformacija dela standardne perspektivne zapremine pogleda između prednje i zadnje ravni odsecanja u standardnu paralelnu zapreminu pogleda.

skraćenja, međutim relativna z-rastojanja se čuvaju, a samim tim i poredak.



Slika 5.17: Standardna perspektivna zapremina pogleda levo (sa prednjom ravni odsecanja $z = -1/4$) sadrži kvadrat koji se transformiše u paralelogram u paralelnoj zapremini pogleda desno, čija duža stranica odgovara manjoj apsolutnoj vrednosti z koordinate.

Ako sa $c = -n/f$ označimo z koordinatu prednje ravni odsecanja nakon transformacije u standardnu perspektivnu zapreminu pogleda, matricu transformacije iz standardne perspektivne zapremine pogleda u standardnu paralelnu zapreminu pogleda (u oznaci M_{pp}) možemo zapisati kao:

$$M_{pp} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/(1+c) & -c/(1+c) \\ 0 & 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & f/(f-n) & n/(f-n) \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Pošto će naknadno biti potrebno homogenizovati koordinate, matrica se može pomnožiti sa $f-n$ i dobiti jednostavnija matrica (koja ne uključuje deljenje):

$$M_{pp} = \begin{bmatrix} f-n & 0 & 0 & 0 \\ 0 & f-n & 0 & 0 \\ 0 & 0 & f & n \\ 0 & 0 & -(f-n) & 0 \end{bmatrix}$$

U ovom materijalu nećemo se baviti izvođenjem ove matrice, ali želimo da se uverimo da ona zaista transformiše zarubljenu piramidu između prednje i zadnje ravni odsecanja standardne perspektivne zapremine pogleda u standardnu paralelnu zapreminu pogleda. To ćemo uraditi proverom uslova za tačke koje odgovaraju krajnjim tačkama perspektivne zapremine pogleda.

Razmotrimo npr. tačku koja odgovara gornjem desnom prednjem uglu standardne perspektivne zapremine pogleda. Ona ima koordinate $(-c, -c, c)$ (podsetimo se da je $c = -n/f$ negativno, pa je vrednost $-c$ pozitivna). Ova tačka se primenom transformacije sa matricom M_{pp} transformiše u tačku sa homogenim koordinatama:

$$\begin{bmatrix} f-n & 0 & 0 & 0 \\ 0 & f-n & 0 & 0 \\ 0 & 0 & f & n \\ 0 & 0 & -(f-n) & 0 \end{bmatrix} \cdot \begin{bmatrix} -c \\ -c \\ c \\ 1 \end{bmatrix} = \begin{bmatrix} -c(f-n) \\ -c(f-n) \\ cf+n \\ -(f-n)c \end{bmatrix}$$

a nakon homogenizacije dobija se:

$$\begin{bmatrix} 1 \\ 1 \\ \frac{cf+n}{-(f-n)c} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

što odgovara gornjem desnom prednjem uglu standardne paralelne zapremine pogleda.

Primetimo da ova transformacija nije afina (poslednja vrsta matrice M_{pp} nije jednaka poslednjoj vrsti jedinične matrice), ali jeste projektivna transformacija. Pošto se pri ovoj transformaciji menja vrednost homogene koordinate, nakon primene matrice transformacije vršićemo homogenizaciju koordinata.

Proces transformisanja proizvoljne (perspektivne ili paralelne) zapremine pogleda u standardnu paralelnu zapreminu pogleda naziva se *normalizacija* (eng. normalization). Procesom normalizacije omogućeno je da i slučaj perspektivnog i paralelnog projektovanja razmatramo na isti način. Važno je istaknuti da se transformisanjem perspektivne zapremine pogleda u paralelnu čuva relativna dubina koja je od ključne važnosti za određivanje vidljivih površina. Dakle, scena će nakon procesa normalizacije imati isti izgled kao i bez primene normalizacije – svako teme biće transformisano na isti način. Cilj procesa normalizacije i jeste da se pojednostave izračunavanja bez izmene onoga što se prikazuje.

Napomenimo na kraju i to da različite grafičke aplikacije očekuju različite opsege z vrednosti: npr. OpenGL očekuje da vrednosti z koordinate budu iz opsega $[-1, 1]$, a Direct3D iz opsega $[0, 1]$. Razmotrimo scenario koji je podržan od strane OpenGL-a – potrebno je preslikati opseg $[0, -1]$ po z osi u opseg $[-1, 1]$. To je moguće postići u dva koraka: skaliranjem sa faktorom -2 po z osi, a zatim translacijom za vrednost -1 duž z ose. Matrica kojom se ovo postiže ima oblik:

$$M_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -2 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Prikazane transformacije kamere omogućavaju da se umesto odsecanja svetskih koordinata primitiva u odnosu na proizvoljnu zapreminu pogleda kamere zadatu svojim parametrima, svetske koordinate transformišu u standardnu paralelnu zapreminu pogleda u kojoj se odsecanja, projektovanje i testovi vidljivosti izvode znatno jednostavnije. U standardnoj paralelnoj zapremini pogleda odsecanja se vrše u odnosu na ravni paralelne koordinatnim osama što odsecanje čini znatno jednostavnijim. Projekcija na ravan slike nije više projekcija na proizvoljnu ravan u trodimenzionom prostoru, već projekcija na ravan paralelnu Oxy ravni u standardnoj paralelnoj zapremini pogleda, što vodi ka tome da je potrebno samo „zaboraviti“ z koordinatu.

Kompletan niz operacija koje se izvršavaju za dati trougao čije su svetske koordinate poznate ima sledeću formu:

1. koordinate temena trougla treba pomnožiti matricom $M_z M_{pp} M_{persp}$ čime se koordinate temena najpre transformišu u koordinate u odnosu na standardnu perspektivnu zapreminu pogleda, a zatim u koordinate u odnosu na standardnu paralelnu zapreminu pogleda;
2. vrši se homogenizacija koordinata tačaka $(x, y, z, w) \rightarrow (x/w, y/w, z/w, 1)$ pri čemu se na homogenu w koordinatu nadalje može zaboraviti;
3. vrši se odsecanje u odnosu na ravni standardne paralelne zapremine pogleda: $x = +/-1$, $y = +/-1$, $z = 0$, $z = -1$;
4. vrši se projektovanje na ravan paralelnu ravni Oxy što odgovara tome da treba samo razmatrati x i y koordinatu tačke (z koordinatu u ovom kontekstu treba odbaciti, mada treba napomenuti da nam je od značaja u kontekstu vidljivosti jer nosi informaciju o dubini potrebnu za test dubine);

5. konačno, dobijene koordinata tačaka transformišu se u odgovarajuće koordinate piksela.

5.4 Pitanja

5.1 Koja je razlika između tačkaste i sintetičke kamere?

5.2 Nabrojati šest parametara sintetičke kamere kod paralelnog i kod perspektivnog projektovanja.

5.3 Šta je frustum?

5.4 Šta se zadaje vektorom pogleda?

5.5 Da li vektor nagore mora biti upravan na vektor pogleda?

5.6 Kako se određuje orijentacija kamere na osnovu vektora pogleda i vektora nagore?

5.7 Šta se zadaje uglovima vidnog polja?

5.8 Kakav je odnos uglova vidnog polja i stepena perspektivnog iskrivljenja na slici?

5.9 Čemu služe prednja i zadnja ravan odsecanja i zašto ih je potrebno zadati? Na koji način se one zadaju?

5.10 Kojim se telom definiše zapremina pogleda kod perspektivnog, a kojim kod paralelnog projektovanja?

5.11 Pokazati da i ovaj pristup daje isti (u, v, w) koordinatni sistem za zadati vektor pogleda i vektor nagore: $w = \frac{-\overrightarrow{look}}{|\overrightarrow{look}|}$, $\overrightarrow{t} = \overrightarrow{w} \times \overrightarrow{up}$, $\overrightarrow{u} = \frac{\overrightarrow{t}}{|\overrightarrow{t}|}$, $\overrightarrow{v} = \overrightarrow{u} \times \overrightarrow{w}$.

5.12 Koji oblik ima standardna perspektivna zapremina pogleda?

5.13 Nacrtati standardnu perspektivnu zapreminu pogleda i napisati koje su njene vrednosti pozicije kamere, vektora pogleda i prednje i zadnje ravni odsecanja.

5.14 Koja je jednačina prednje ravni odsecanje kod standardne perspektivne zapremine pogleda?

5.15 Kako izgleda standardna paralelna zapremina pogleda?

5.16 Koje su vrednosti pozicije kamere, vektora pogleda, vektora nagore i prednje i zadnje ravni odsecanja u slučaju: (a) standardne perspektivne zapremine pogleda (b) standardne paralelne zapremine pogleda?

5.17 Iz kog razloga se vrši transformacija iz proizvoljne perspektivne u standardnu perspektivnu zapreminu pogleda?

5.18 Zašto se vrši transformacija iz perspektivne u paralelnu zapreminu pogleda?

Glava 6

Vidljivost

Jedan od fundamentalnih problema u računarskoj grafici je da se za dati piksel utvrdi koji se fragment sa scene preslikava u njega. On može biti veoma složen i vremenski zahtevan u slučaju scena koje sadrže veliki broj objekata. Ispitivanje vidljivosti svake pojedinačne primitive na sceni bilo bi vremenski izuzetno skupo i neprihvatljivo u primenama koje treba da rade u realnom vremenu. Takođe, mala promena tačke pogleda može da prouzrokuje velike promene u vidljivosti.

Ukoliko ne bi postojao razvijen mehanizam za utvrđivanje koje površi na sceni su vidljive iz pozicije kamere, vidljivost površi zavisila bi od redosleda u kojem su one iscrtavane, tako što bi one kasnije iscrtane zaklonile one iscrtane ranije. Neophodno je izbegnuti situaciju u kojoj bi korisnik trebalo da vodi računa o redosledu iscrtavanja poligona, već bi bilo poželjno da se grafičkoj funkciji proslede poligoni koje treba iscrtati, a da ona sama utvrdi koji su poligoni ispred kojih u odnosu na kameru.

Problem utvrđivanja koje su površi vidljive nazivamo *određivanjem vidljivih površi* (eng. visible surface determination – VSD), a nekad se naziva i *odbacivanje skrivenih površi* (eng. hidden surface removal – HSR).

Problem određivanja vidljivih površi je složen, te je s jedne strane potrebno obezbediti tačnost rezultata koje algoritam vraća, a s druge strane konstantno raditi na povećanju njegove efikasnosti. Algoritmi određivanja vidljivih površi koji garantuju korektnost rezultata moraju da potpuno precizno utvrde da li između dve tačke postoji linija vidljivosti na kojoj nema prepreka. Osnovna primena ovakvog tipa algoritma je određivanje *primarne vidljivosti* (eng. primary visibility, camera visibility), odnosno utvrđivanje koje su površi vidljive iz pozicije kamere. Na ovaj način se iscrtavaju samo delovi scene koji su zaista vidljivi iz pozicije kamere i garantuje se ispravan rezultat renderovanja. Danas najpopularniji algoritmi ovog tipa su rej kasting algoritam i z-bafer algoritam. Drugu vrstu algoritama za određivanje vidljivih površi čine algoritmi konzervativnog određivanja vidljivosti čiji je cilj da što efikasnije izdvoje delove scene koji sigurno nisu vidljivi iz oka posmatrača, dok je na preostale objekte potrebno primeniti neki od algoritama koji garantuju tačnost rezultata koji vraćaju.

Saderland je još 1973. godine podelio algoritme za utvrđivanje vidljivih površi na:

- *algoritme prostora objekta* (eng. object precision algorithms) – koji rade direktno nad objektima u svetskom koordinatnom sistemu i
- *algoritme prostora slika* (eng. image precision algorithms) – koji rade nad projekcijama objekata u koordinatama ekrana, a vraćaju se u svetski koordinatni sistem onda kada je potrebna vrednost z koordinate.

Cilj prve klase algoritama je da se za svaki objekat odredi koji njegov deo je vidljiv; to se postiže na taj način što se objekti međusobno porede i eliminišu se kompletni objekti ili delovi objekta koji nisu vidljivi (slika 6.1). Kod ove vrste algoritama rezolucija uređaja za prikaz nije relevantna jer se izračunavanja izvode na nivou objekata. Složenost ovih algoritama je u najgorem slučaju $O(n^2)$, gde je n broj objekata.

Algoritmima prostora slike se pak za svaki piksel određuje koji od objekata je vidljiv: za svaki piksel utvrđuje se koji je objekat najbliži posmatraču i piksel se boji bojom najbližeg objekta. Dakle, algoritmi ovog tipa nastupaju nakon što se objekti konvertuju u niz piksela u kolor baferu. Rezolucija prikaznog uređaja je u ovom slučaju od značaja jer se poređenja rade na nivou piksela (slika 6.2). Vremenska složenost ovih algoritama u najgorem slučaju iznosi $O(p \cdot n)$, gde je sa p označen ukupan broj piksela na slici, a sa n broj objekata.

```
za svaki objekat na sceni:
   utvrdi koji delovi objekta su vidljivi
   (ukljucuje poredjenje poligona objekta A
    sa ostalim poligonima objekta A
    i sa poligonima ostalih objekata na sceni)
```

Slika 6.1: Pseudokod algoritama prostora objekta.

```
za svaki piksel u frejm baferu:
   utvrdi koji poligon je najblizi posmatracu na toj lokaciji
   oboji piksel bojom tog poligona na toj lokaciji
```

Slika 6.2: Pseudokod algoritama prostora slike.

Danas najveći broj algoritama za utvrđivanje vidljivih površi pripada grupi algoritama prostora slike i neki od najpopularnijih su z-bafer algoritam i rej kasting algoritam.

6.1 Algoritmi konzervativnog određivanja vidljivosti

Za potrebe efikasnijeg određivanja vidljivih površi razvijeni su *algoritmi konzervativnog određivanja vidljivosti* (eng. conservative visibility algorithms). Oni su posebno važni u situacijama kada se vrši renderovanje u realnom vremenu jer složene scene mogu da sadrže mnogo više primitiva nego što raspoloživi hardver može da obradi, a može se desiti da je samo mali procenat tih primitiva vidljiv iz pozicije kamere. Ideja konzervativnih algoritama je da se razdvoje delovi scene koji su verovatno vidljivi od delova scene koji sigurno nisu vidljivi. Eliminacija primitiva koje sigurno nisu vidljive smanjuje broj potrebnih testova vidljivosti, ali sama po sebi ne garantuje korektnost jer ne moraju svi preostali objekti biti vidljivi. S obzirom na to da se ovakvim metodama rezultat dobija mnogo brže nego algoritmima koji garantuju tačnost, renderovanje se može ubrzati na sledeći način: najpre se primeni jedan ili veći broj brzih konzervativnih metoda koji sužavaju prostor primitiva koji se razmatra odbacivanjem primitiva za koje se jednostavno može utvrditi da neće biti vidljive, a zatim se na taj manji preostali skup primitiva primeni neki od (vremenskih zahtevnijih) algoritama koji garantuje korektnost. Na primer, veoma efikasno se može utvrditi da se sfera koja ograničava mrežu trouglova nalazi iza kamere i stoga nije vidljiva iz pozicije kamere, dok bi zasebna provera vidljivosti svakog od trouglova mreže mogla biti vremenski veoma zahtevna.

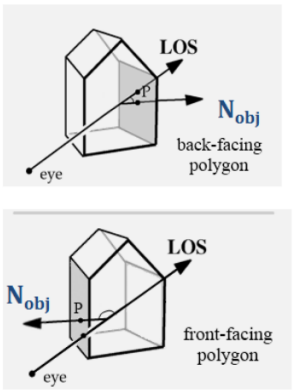
Tri osnovna algoritma konzervativnog određivanja vidljivosti su *odbacivanje dela van zarubljene piramide pogleda* (eng. frustum culling), *odbacivanje zadnje strane (naličja) objekta* (eng. backface culling) i *blokirajuće odbacivanje* (eng. occlusion culling). Prva dva algoritma spadaju u jednostavnije, dok je blokirajuće odbacivanje složenije i u obzir uzima uzajamno blokiranje između objekata na sceni. Algoritmi konzervativnog testiranja vidljivosti se mogu učiniti vremenski efikasnijim korišćenjem pogodnih prostornih struktura podataka, o kojima će biti reč u kasnijim poglavljima.

Slika 6.3: Odbacivanje u odnosu na zapreminu pogleda.

Odbacivanje u odnosu na zapreminu pogleda svodi se na to da se za svako teme poligona izvrši provera da li se nalazi sa druge strane ravni koje ograničavaju zapreminu pogleda (slika 6.3). Posebno

je korisno u situacijama kada se na sceni nalazi veliki broj složenih modela, a samo mali deo njih je vidljiv iz pozicije kamere. Ovaj metod se nekada može unaprediti tako što se oko svakog složenog modela opiše neka jednostavna figura (sfera ili kocka čije su strane paralelne koordinatnim ravnima) i izvrši provera da li se ona nalazi unutar zapremine pogleda: ukoliko to nije slučaj, može se trivijalno odbaciti celokupna složena figura sadržana u njoj.

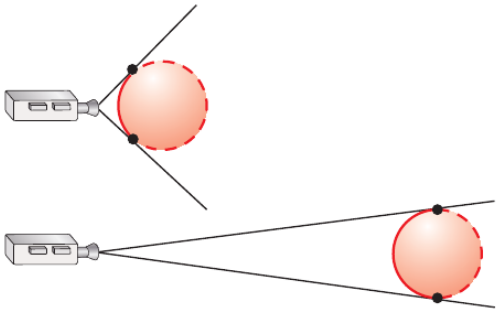
Primetimo da kada radimo sa neprozirnim čvrstim telima, zadnja strana objekta neće biti vidljiva. Odbacivanje primitiva koje se nalaze sa zadnje strane objekta može da eliminiše oko polovine geometrije scene. Na primer, ako razmatramo model kocke, najviše tri strane kocke biće vidljive, odnosno najmanje tri od šest strana kocke se nalaze sa zadnje strane objekta u odnosu na posmatrača i mogu se eliminisati iz daljeg razmatranja.



Slika 6.4: Određivanje prednje i zadnje strane objekta.

Zadnju stranu objekta možemo geometrijski identifikovati na sledeći način. Pretpostavimo da imamo zatvorenu mrežu poligona bez samopreseka i kameru koja se nalazi u tački izvan poliedra definisanog mrežom (slika 6.4). Neka je P tačka poligona čiju vidljivost ispitujemo i $\overrightarrow{N}$ normala ravni kojoj dati poligon pripada usmerena ka spoljašnjosti. Kažemo da je poligon sa prednje strane u odnosu na kameru ako se kamera nalazi u pozitivnoj poluravni datog poligona, a da je sa zadnje strane ako se nalazi u negativnoj poluravni poligona. Ako se kamera baš nalazi u ravni poligona, tada se poligon nalazi na konturi koja razdvaja prednju od zadnje strane objekta. Ako sa $\overrightarrow{LOS}$ (eng. line of sight) označimo vektor pogleda kamere, ispitivanje da li je poligon sa prednje ili zadnje strane u odnosu na poziciju kamere može se svesti na određivanje znaka skalarnog proizvoda vektora $\overrightarrow{LOS}$ i vektora normale $\overrightarrow{N}$ poligona (usmerenog ka spoljašnjosti):

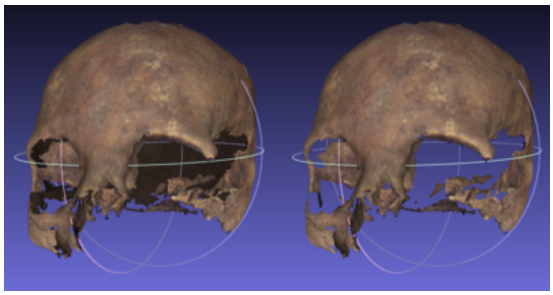
- ako je $\overrightarrow{LOS} \cdot \overrightarrow{N} < 0$ poligon je sa prednje strane u odnosu na kameru,
- ako je $\overrightarrow{LOS} \cdot \overrightarrow{N} > 0$ poligon je sa zadnje strane u odnosu na kameru,
- ako je $\overrightarrow{LOS} \cdot \overrightarrow{N} = 0$ poligon je na konturi koja razdvaja prednju od zadnje strane.



Slika 6.5: Prva sfera ima veću zadnju stranu od druge jer je bliža kameri.

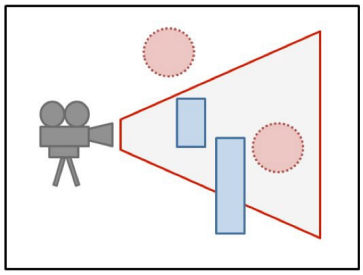
Primetimo da objekat bliži kameri teži tome da ima veću zadnju stranu od objekta koji se nalazi dalje od kamere (slika 6.5).

Odbacivanje zadnje strane objekta možemo izvršiti samo u slučaju zatvorenog neprozirnog objekta. Kod objekata koji imaju "rupe" mogu se videti i poligoni koji pripadaju zadnjoj strani objekta (slika 6.6).



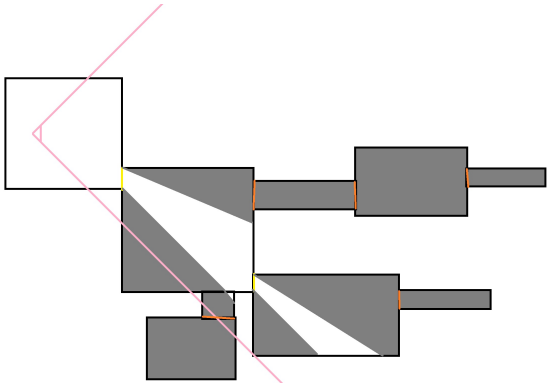
Slika 6.6: Razlika između modela u slučaju kada je izvršeno odbacivanje zadnje strane objekta i kada nije izvršeno (preuzeto sa https://en.wikipedia.org/wiki/Back-face_culling).

Objekte koji se u potpunosti nalaze iza neprozirnih objekata možemo odbaciti iz daljeg razmatranja i ovaj mehanizam nazivamo blokirajuće obacivanje (slika 6.7). Ovo je veoma popularan mehanizam za ubrzavanje renderovanja složenih scena.



Slika 6.7: Ilustracija odbacivanja u odnosu na zapreminu pogleda i blokirajućeg odbacivanja – gornja sfera nije vidljiva jer se nalazi van zapremine pogleda, dok donja sfera nije vidljiva jer je zaklonjena drugim netransparentnim objektima (preuzeto sa https://medium.com).

Postoji veći broj tipova blokirajućeg odbacivanja: jedno od njih je hijerarhijsko odbacivanje koje koristi hijerarhijsku reprezentaciju scene i vodi se principom da složeni objekat nije vidljiv ukoliko se pokaže da jednostavnija geometrijska figura opisana oko objekta nije vidljiva. Drugi tip blokirajućeg odbacivanja posmatra scenu izdeljenu na veći broj ćelija (soba) koje su povezane putem portala (vrata). Portali su prozori u trodimenzionom prostoru koji omogućavaju da se objekti koji se nalaze u jednoj ćeliji vide iz druge ćelije. Portali su najprimenljiviji u razmatranju unutrašnjih prostora, odnosno u scenama kod kojih postoji ograničena vidljivost iz jednog prostora u drugi i imaju ograničenu primenu u otvorenim scenama. Ideja korišćenja ćelija i portala jeste ta što nije potrebno iscrtavati geometriju u susednoj ćeliji ako su vrata zatvorena. Ukoliko su vrata otvorena, onda je potrebno iscrtati samo geometriju koja je vidljiva kroz ta vrata (slika 6.8).



Slika 6.8: Ilustracija blokirajućeg odbacivanja u slučaju kada je prostor predstavljen u vidu ćelija i portala.

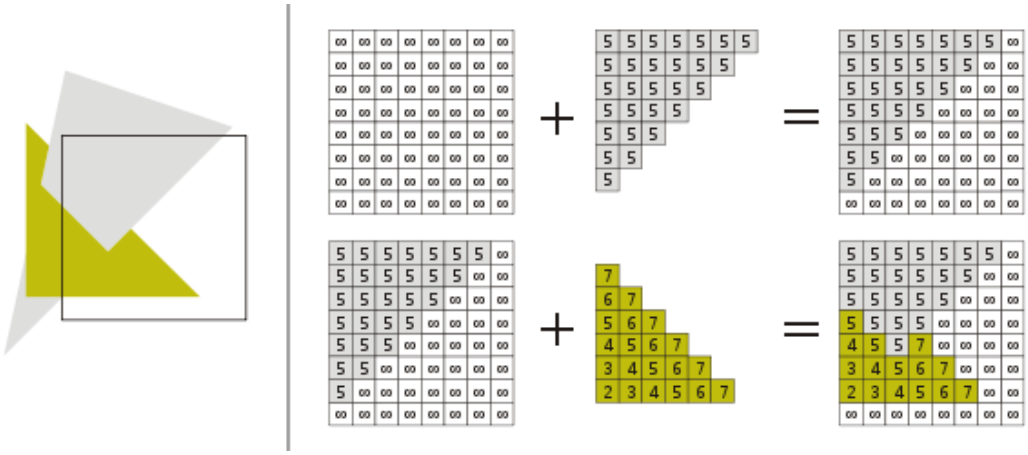
Jedna od važnih već pomenutih tehnika koja može dovesti do efikasnijih algoritama jeste neprekidnost, odnosno koherentnost, koja dolazi u različitim oblicima. Naime, svojstva objekta (kao što su njegova geometrija, boja, osvetljenje i sl.) se obično menjaju glatko i postoji određena sličnost vrednosti ovih svojstava u bliskim tačkama. Neprekidnost omogućava da se izračunavanja koja su izvedena za jedan deo objekta ponovo iskoriste u neizmenjenom obliku ili sa nekim malim inkrementalnim izmenama za bliske delove objekta. Na primer, neprekidnost površi podrazumeva da će susedni pikseli vrlo verovatno biti pokriveni istom stranom objekta, odnosno ivica menja svoju vidljivost samo ako preseče neku drugu vidljivu ivicu ili ako prodire kroz vidljivu stranu. Dodatno, svojstva površi će se vrlo verovatno postepeno menjati, te se za izračunavanja za jednu stranu objekta može iskoristiti tehnika inkrementalnosti. Slično, kada se izračuna dubina jedne tačke date površi, dubine susednih tačaka te strane imaće sličnu vrednost koja se može dobiti inkrementalnom metodom.

6.2 z-bafer algoritam

Grafička protočna obrada za problem rasterizacije obrađuje svaki trougao zasebno i nije svesna postojanja drugih trouglova. U ovakvom kontekstu postavlja se pitanje na koji način onda iskombinovati informacije dobijene za svaki pojedinačni trougao. Ovaj problem se može rešiti održavanjem informacije o trenutnoj vidljivosti putem *z-bafera*, odnosno *bafera dubine* (eng. depth buffer). Naime, z-bafer u najjednostavnijoj varijanti za svaki piksel slike čuva jednu skalarnu vrednost koja daje neku meru rastojanja centra projekcije od površi čijom se bojom boji taj piksel. Kako bi se redukovao aliasing efekat koji se dobija korišćenjem jedne vrednosti po pikselu, često se za svaki piksel čuva više uzoraka boje i dubine. U daljem tekstu jednostavnosti radi podrazumevaćemo da se za svaki piksel čuva samo po jedna vrednost dubine.

Bafer dubine omogućava određivanje informacije o vidljivosti tokom rasterizacije. Dok se scena renderuje, bafer dubine sadrži implicitnu informaciju o utvrđivanju vidljivih površi: nova površ može da pokrije trenutnu vrednost u kolor baferu samo ako je njena dubina u odnosu na poziciju kamere manja od one sadržane u baferu dubine. Ako je tako, nova boja se upisuje na mesto stare u kolor baferu, a nova vrednost dubine u bafer dubine. Ovo je tzv. implicitna vidljivost jer sve dok se ne završi renderovanje kompletne scene nije poznato koja je najbliža vidljiva površ. Kada se proces renderovanja završi, obezbeđena je korektna informacija o vidljivosti.

z-bafer algoritam (eng. z-buffer algorithm, depth-buffer algorithm), koji je 1974. godine predložio Edvin Katmul (Edwin Catmull)[1], predstavlja jedan od najjednostavnijih algoritama za određivanje vidljivih površi. Ovaj algoritam pripada grupi algoritama prostora slika. Kao što i sam naziv algoritma sugeriše, u algoritmu se pretpostavlja postojanje i intenzivno korišćenje z-bafera u kome se čuvaju vrednosti z koordinate za svaki piksel.



Slika 6.9: Ilustracija z-bafer algoritma. Manje vrednosti zamenjuju veće u z-baferu.

Na početku algoritma sve vrednosti u kolor baferu inicijalizuju se na boju pozadine, dok se sve vrednosti u z-baferu inicijalizuju na z koordinatu zadnje ravni odsecanja. Maksimalna dopuštena vrednost u z-baferu je z koordinata prednje ravni odsecanja (kao maksimalna vrednost z koordinate koja pripada zapremini pogleda).

[1] Edvin Katmul je dugi niz godina bio predsednik Piksara i Volt Dizni studija, a dobitnik je i Tjuringove nagrade za svoje doprinose u oblasti računarske grafike.

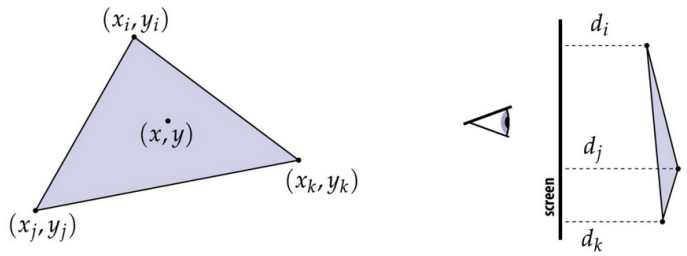
Na sve poligone kojima je predstavljena scena primenjuje se algoritam rasterizacije. Redosled u kome se obrađuju poligoni je proizvoljan. Prilikom procesa rasterizacije, ako je dubina d tačke poligona na sceni koja se rasterizuje u piksel (x, y) manja od vrednosti koja se trenutno nalazi u z-baferu za dati piksel, onda se u z-bafer za piksel (x, y) upisuje dubina d, a u kolor bafer boja tačke poligona koji se rasterizuje u piksel (x, y). Primetimo da z-bafer radi ispravno i u situaciji kada se površi međusobno presecaju. Naime, to što je neki trougao ispred drugog u jednom svom delu, a iza njega u svom drugom delu ne predstavlja problem, jer se dubine porede u tačkama uzorkovanja, a ne uopšteno. Na slici 6.10 prikazan je pseudokod z-bafer algoritma.

```
Procedure zBuffer
var
 pz: integer;
begin
 for y := 0 to YMAX do
  for x := 0 to XMAX do
   begin
    WritePixel(x,y,background_value);
    WriteZ(x,y,1)
   end

 za svaki poligon uradi sledece
  za svaki piksel u projekciji poligona uradi sledece
   begin
    pz := z-vrednost poligona u tacki sa koordinatama (x,y);
    if pz <= ReadZ(x,y) then
     begin { ova tacka nije dalja  }
      WriteZ(x,y,pz);
      WritePixel(x,y,boja_poligona_u_tacki(x,y))
     end
   end
end.
```

Slika 6.10: z-bafer algoritam.

Primetimo da je u ovom algoritmu implicitno potrebno rešiti problem izračunavanja z koordinate, odnosno dubine proizvoljne tačke poligona, ako su nam poznate vrednosti z-koordinata njegovih temena. Pokazuje se da se recipročna vrednost dubine menja linearno duž trougla, te je dubinu proizvoljne tačke poligona moguće izračunati interpolacijom recipročnih vrednosti dubine temena, korišćenjem baricentričnih koordinata tačke (slika 6.11).



Slika 6.11: Problem određivanja z-koordinate (dubine) tačke iz unutrašnjosti trougla, kada su z-koordinate temena različite.

Prednost z-bafer algoritma je što pre primene algoritma nije potrebno sortirati objekte i nije potrebno međusobno porediti objekte.

Primetimo da z-bafer algoritam ne zahteva nužno da primitivne površi budu trouglovi: on se može iskoristiti za renderovanje proizvoljnog objekta ako se boja i z vrednost mogu utvrditi za svaku tačku

u njegovoj projekciji. Na primer, moguće je na ovaj način crtati sferu jer je jednostavno odrediti njenu dubinu u svakoj pojedinačnoj tački.

z-bafer algoritam se veoma jednostavno implementira. Zbog svoje jednostavnosti, može se implementirati i u hardveru. Podaci smešteni u z-baferu se mogu čuvati zajedno sa kolor baferom čime se omogućava da se toj sceni jednostavno naknadno doda novi objekat.

Ipak, pri realizaciji z-bafer algoritma treba biti svestan i nekih potencijalnih problema. Naime, z-bafer ima konačnu preciznost, te ako dva fragmenta imaju bliske vrednosti dubine, oni se mogu preslikati u iste vrednosti u z-baferu te nećemo znati koji treba da bude iscrtan ispred kog. Ovo se naime dešava i češće nego što bismo pretpostavili i posledica je delovanja matrice transformacije standardne perspektivne u standardnu paralelnu zapreminu pogleda. Podsetimo se izgleda ove matrice:

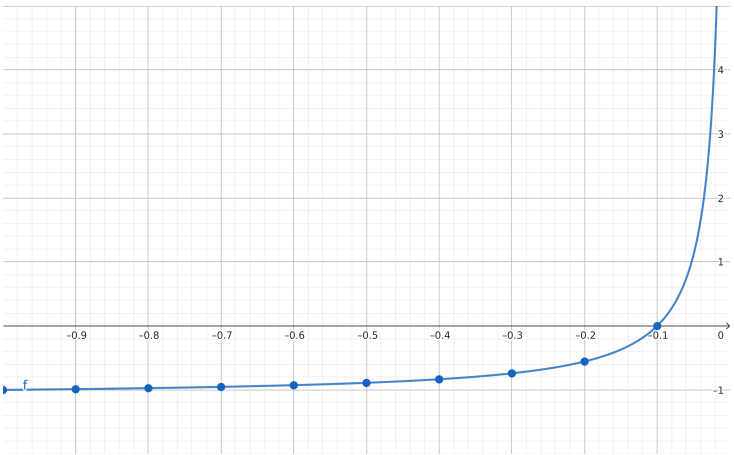
$$M_{pp} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/(1+c) & -c/(1+c) \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

gde je $c = -n/f$ i razmotrimo na koji način matrica M_{pp} deluje na neku tačku:

$$M_{pp} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/(1+c) & -c/(1+c) \\ 0 & 0 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ \frac{z-c}{1+c} \\ -z \end{bmatrix}$$

Koordinate ove tačke će nakon homogenizacije biti jednake $\begin{bmatrix} -x/z & -y/z & \frac{c-z}{z+zc} & 1 \end{bmatrix}^T$. Kao što bismo i očekivali, sa povećanjem z koordinate veći je stepen perspektivnog skraćenja po x i y koordinati. Označimo novu vrednost dubine (tj. z-koordinate) $\frac{c-z}{z+zc}$ sa z' i razmotrimo kakva je njena vrednost u odnosu na polaznu z-koordinatu tačke. Pokazuje se da se pri transformaciji standardne perspektivne u standardnu paralelnu zapreminu pogleda, tačke kompresuju ka zadnjoj ravni odsecanja, odnosno ka vrednosti $z' = -1$. Ovaj efekat naziva se *z-kompresija* (eng. z-compression). Naime, što je prednja ravan odsecanja bliža ravni $z = 0$, a zadnja ravan odsecanja udaljenija, vrednost $c = -n/f$ biće bliskija vrednosti 0, a efekat z-kompresije je izraženiji (slika 6.12). Algebarski razmatrano, važi:

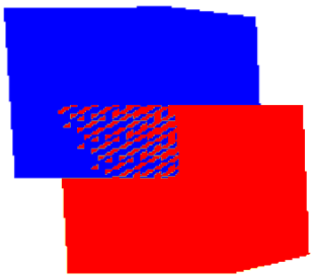
$$z' = \frac{c-z}{z+zc} \approx -\frac{z}{z} = -1$$



Slika 6.12: Ilustracija funkcije $z' = \frac{c-z}{z+zc}$ za $n = 0.1$ i $f = 1$, tj. za vrednost $c = -0.1$.

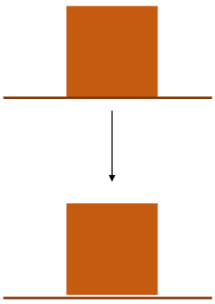
Efekat z-kompresije se vizualno manifestuje kao *z-konflikt* (eng. z-fighting), odnosno kao pojava da se dve primitive "bore" koja će primitiva biti ispred koje (slika 6.13). Posebno je problematična situacija kada se kamera kreće i onda se tokom vremena menja informacija o tome koja od ove dve primitive je vidljiva. Dodatno, ovaj efekat je jači kada su objekti na većem rastojanju jer za veće vrednosti z-bafer ima manju preciznost.

Razmotrimo na koji način je moguće postići da se ovaj efekat javlja što je moguće ređe. Najpre je, koliko je to moguće, potrebno izbegavati da se objekti postavljaju preblizu jedan drugome. Ukoliko je pak neophodno da se objekti preklapaju, moguće je između njih veštački napraviti mali razmak koji



Slika 6.13: Ilustracija z-konflikta.

je za korisnika neprimetan, a koji bi omogućio jednostavniji rad z-bafer algoritma (slika 6.14). Takođe, sa povećanjem preciznosti bafera dubine ovaj efekat će se ređe javljati. Na kraju, poželjno je povećati vrednost razlomka n/f, tako što ćemo zadnju ravan odsecanja primaknuti kameri, a prednju ravan odsecanja odmaknuti od kamere.



Slika 6.14: Eliminisanje preklapanja primitiva dodavanjem malog razmaka između njih.

6.3 Rej kasting algoritam

Rej kasting algoritam (eng. ray casting algorithm) utvrđuje da li je neka površ vidljiva praćenjem imaginarnog zraka svetlosti od pozicije kamere do objekta na sceni. Pretpostavimo da je prozor proizvoljne ravni pogleda podeljen pravilnom mrežom tako da elementi odgovaraju pikselima potrebne rezolucije. Zamislimo, takođe, da pustimo zrak svetlosti kroz centar piksela tog prozora – objekat koji zrak pogodi je onaj koji treba da bude prikazan na poziciji tog piksela. Dakle, za zadatu poziciju kamere i prozor ravni pogleda za svaki piksel se prati po jedan zrak od centra projekcije kroz sam piksel do najbližeg objekta na sceni (slika 6.15). Boja piksela se postavlja na boju najbližeg objekta u pravcu odgovarajućeg zraka. Rej kasting algoritam pripada grupi algoritama prostora slika.

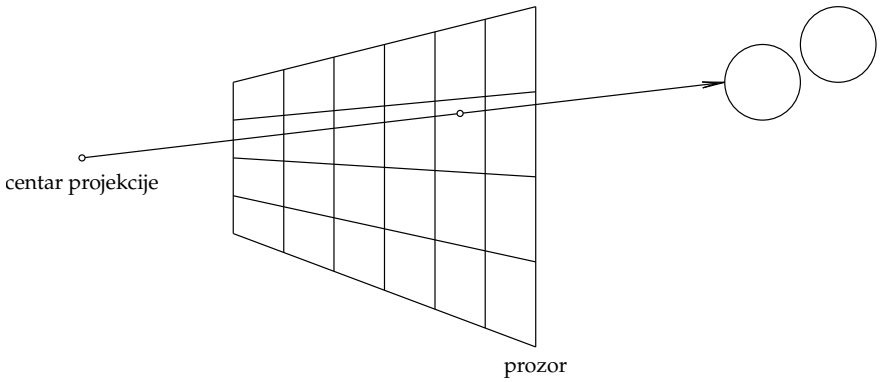
Prva tačka preseka zraka i primive scene predstavlja rezultat tzv. *upita preseka* (eng. intersection query). Primetimo da rej kasting algoritam predstavlja direktan proces odgovaranja na upit preseka.

Pseudokod rej kasting algoritma prikazan je na slici 6.16.

Naziv rej kasting se obično koristi za varijantu algoritma kojom se vrši samo određivanje vidljivosti, dok se za puni rekurzivni algoritam kojim se obrađuju i senke, refleksija i prelamanje koristi naziv *rej trejsing* (eng. ray tracing).

Rej trejsing algoritam su razvili Artur Apel (Arthur Appel) 1968. godine i Robert Goldštajn (Robert Goldstein) i Rodžer Nagel (Roger Nagel) 1971. godine. Apel je do algoritma došao rešavajući problem ispitivanja da li je tačka u senci, dok su ga Goldštajn i Nagsel originalno razvili za simuliranje putanja balističkih projektila, pa su tek naknadno uvideli da on ima primenu u računarskoj grafici.

U osnovnoj verziji rej kasting algoritma za svaki piksel (x_1, y_1, z_1) potrebno je ispitati da li postoji presečna tačka zraka iz centra projekcije (x_0, y_0, z_0) sa svim objektima na sceni, tj. potrebno je odrediti vrednost parametra t tako da tačka sa koordinatama $x = x_0 + t(x_1 - x_0)$, $y = y_0 + t(y_1 - y_0)$, $z = z_0 + t(z_1 - z_0)$ pripada objektu. Pritom sva izračunavanja treba da budu što je moguće brža. Za efikasnu primenu rej kasting algoritma koriste se mnoge optimizacije, kao što je npr. optimizacija izračunavanja preseka. Naime, prilikom računanja preseka neki izrazi su konstantni za celu sliku ili su konstantni za dati objekat, pa se mogu izračunati unapred i koristiti u daljem izračunavanju (na primer, kod ortografske projekcije poligona na ravan). U idealnom slučaju, svaki zrak bi trebalo testirati na



Slika 6.15: Ilustracija rej kasting algoritma.

```
izaberi centar projekcije i prozor na projekcionoj ravni;
za svaku sken liniju na slici uradi sledece:
  za svaki piksel na sken liniji uradi sledece:
    begin
      odredi polupravu iz centra projekcije kroz piksel;
      za svaki objekat na sceni uradi sledece:
        ako poluprava sece objekat i
        ako je presecna tacka najbliza do sada onda
          zapamti presek i ime objekta;
      oboji piksel bojom najblizeg preseka
    end
```

Slika 6.16: Rej kasting algoritam.

presek samo sa objektima koje zaista i preseca. U ove svrhe se mogu iskoristiti neprekidnost zraka i neprekidnost primitiva, sa idejom da susedni zraci presecaju sličan skup objekata i da su susedni pikseli najčešće "pokriveni" istom stranom objekta. Još bi bolje bilo kada bi se izračunao presek samo sa objektom koji je najbliži početku zraka. Razvijene su različite tehnike koje pokušavaju da postignu ovaj cilj i one uključuju korišćenje hijerarhija i prostorno particionisanje. Na primer, ako je zadat objekat koga je skupo testirati na presek (sastoji se od velikog broja poligona), on se može umetnuti u neki jednostavni objekat kao što su sfera ili kvadar za koji je ispitivanje preseka jednostavno. Ukoliko zrak ne seče objekat u koji je umetnut, onda on ne seče ni dati objekat. Na ovoj ideji formiraju se kompletne hijerarhije graničnih opsega koje pojednostavljuju rešavanje upita preseka.

Složenost rej kasting algoritma za jedan zrak u odnosu na n primitiva u nizu je $O(n)$. Linearna složenost u funkciji broja primitiva je nepraktična za velike i složene scene, posebno imajući u vidu da baš kod takvih scena često veliki broj površi nije vidljiv iz date tačke. Dakle, rej kasting algoritam je jednostavan i elegantan, ali je vremenski veoma zahtevan. Međutim, njegova vremenska složenost se može poboljšati korišćenjem naprednih struktura podataka.

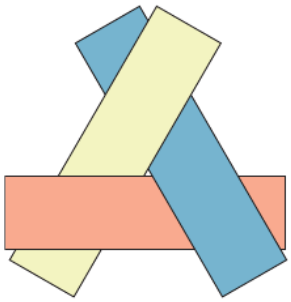
6.4 Algoritmi sa listama prioriteta

Algoritmi sa listama prioriteta implicitno rešavaju problem vidljivosti tako što renderuju elemente scene u redosledu u kom udaljeniji objekti imaju viši prioritet te se prvi renderuju i bivaju naknadno zaklonjeni bližim objektima koji se kasnije renderuju. Ovi algoritmi su ranije imali značajnu primenu, korišćeni su recimo u prvom simulatoru letenja koji je računarski generisao slike u realnom vremenu, a koji je razvijao General Electric. U današnje vreme se ovaj tip algoritama gotovo ne koristi jer su razvijene bolje alternative. Ipak, jednostavnost implicitnog utvrđivanja vidljivosti zadavanjem prioriteta čini ih korisnim za neke specijalne namene.

6.4.1 Slikarev algoritam

Razmotrimo proces kojim umetnik slika pejzaž. Slikar najpre boji nebo, a preko njega slika planine koje zaklanjaju nebo. U prednjem planu, slikar slika drveće preko planina, a zatim kuću preko drveta. Ovaj proces se može nastaviti i on se u računarskoj grafici naziva *slikarevim algoritmom* (eng. Painter's algorithm). Zaklanjanje objekata i utvrđivanje vidljivosti se postižu tako što se boje daljih tačaka "pregaze" bojama bližih tačaka. Ova ideja se može primeniti na svaki pojedinačni piksel jer za svaki piksel postoji korektno uređenje tačaka od najdalje ka najbližoj koje direktno utiču na njega. Međutim, ovakva varijanta algoritma bila bi veoma neefikasna jer bi zahtevala sortiranje svih tačaka svakog od objekata.

Kako bi se vremenska složenost popravila, slikarev algoritam se često primenjuje ne na pojedinačne piksele već na primitive u celini, npr. trouglove mreže. U ovom slučaju algoritam u opštem slučaju ne uspeva. Naime, primitive veće od tačaka se nekada ne mogu urediti od najudaljenije do najbliže (slika 6.17). Ako bismo pak dozvolili podelu primitiva tamo gde se njihove projekcije seku mogli bismo dobiti ispravno uređenje primitiva.



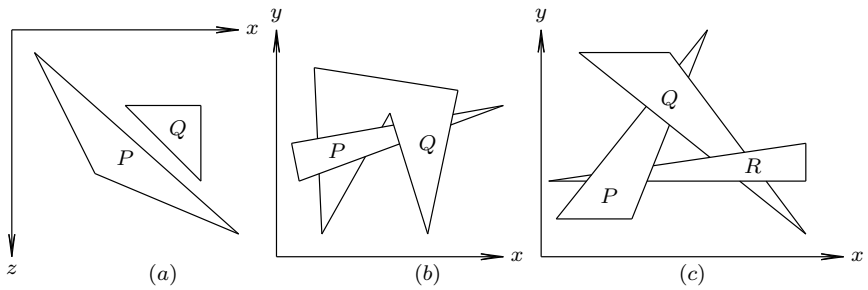
Slika 6.17: Tri konveksna poligona koja se ne mogu ispravno renderovati korišćenjem slikarevog algoritma zbog njihovog uzajamnog preklapanja. Za svaku tačku postoji striktno uređenje po dubini, ali ne postoji korektno uređenje kompletnih poligona.

6.4.2 Algoritam sortiranja dubine

Braća Martin i Dik Njuel (Martin Newell, Dick Newell) su zajedno sa Tomom Sanča (Tom Sancha) 1972. godine predložili algoritam koji predstavlja uopštenje slikarevog algoritma na poligone tako da se u svim slučajevima dobije ispravan rezultat. Ovaj algoritam poznat je pod nazivom *algoritam sortiranja dubine* (eng. depth-sort algorithm). Ravan kojoj pripada poligon ne mora biti paralelna Oxy ravni (odnosno ne moraju sve tačke tog poligona imati istu vrednost z koordinate) pa je uobičajeno da se najpre poligoni grubo sortiraju prema najmanjoj (najudaljenijoj) z koordinati poligona, da se nakon toga izvrši rasterizacija najudaljenijeg poligona, a zatim napreduje ka poligonima koji su bliži kameri. Preciznije, algoritam sortiranja dubine se sastoji iz četiri koraka:

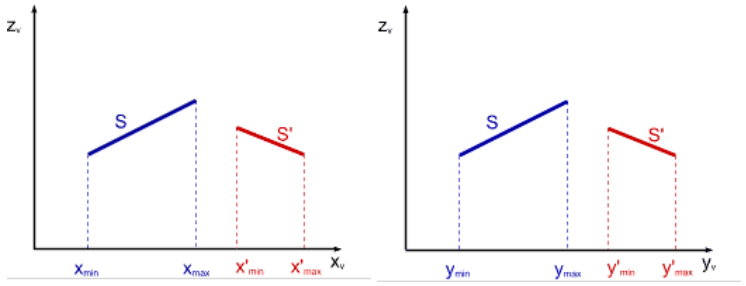
1. svakom poligonu se dodeljuje ključ sortiranja koji je jednak z vrednosti temena koje je najdalje od oblasti za prikaz,
2. poligoni se sortiraju od najudaljenijeg do najbližeg prema vrednosti ključa,
3. utvrđuju se slučajevi kada dva poligona imaju dvosmisleno uređenje – u tom slučaju potrebno je deliti ovakve poligone sve dok njihovi delovi nemaju eksplicitno uređenje i postaviti ih na pravo mesto u sortiranoj listi,
4. poligoni se renderuju u redosledu prioriteta, od najudaljenijeg do najbližeg.

Poligoni se obrađuju jedan po jedan od najudaljenijeg ka najbližem. Ukoliko se objekti međusobno ne preklapaju po dubini (z koordinati) onda je jedino potrebno renderovati ih od najudaljenijeg ka najbližem. Nažalost, to često nije slučaj te traženo uređenje ne mora da postoji. U tom slučaju potrebno je izvršiti dodatne testove. Za tekući poligon P, koji je najdalji od posmatrača i nalazi se na kraju uređene liste poligona, potrebno je ispitati svaki poligon Q čije se z koordinate preklapaju sa z koordinatama poligona P, kako bismo bili sigurni da poligon P ne može da zakloni Q i da se stoga može iscrtati pre poligona Q. Na poligone P i Q primenjuje se lista od narednih pet testova, uređena u rastućem redosledu složenosti:

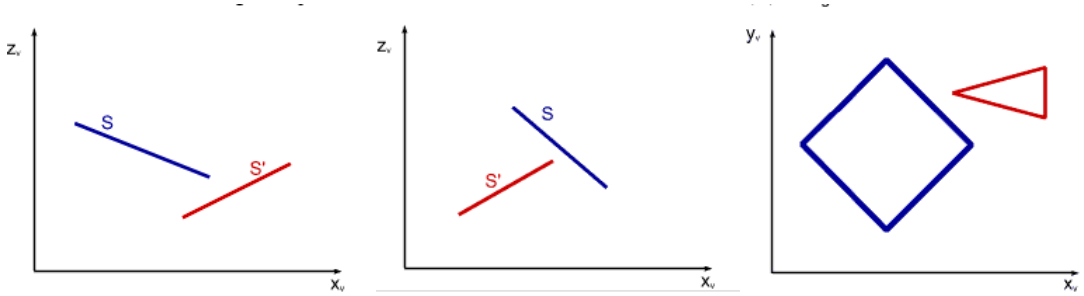


Slika 6.18: Različiti položaji poligona.

1. da li se x koordinate dva poligona ne preklapaju (tj. ne postoje tačke poligona P i Q sa istom x koordinatom)?
2. da li se y koordinate dva poligona ne preklapaju (tj. ne postoje tačke poligona P i Q sa istom y koordinatom)?
3. da li je čitav poligon P sa suprotne strane ravni poligona Q u odnosu na posmatrača?
4. da li je čitav poligon Q sa iste strane ravni poligona P kao i posmatrač?
5. da li su projekcije poligona P i Q na ravan projekcije disjunktne?



Slika 6.19: Ilustracija testova 1 i 2 u algoritmu sortiranja dubine.



Slika 6.20: Ilustracija testova 3, 4 i 5 u algoritmu sortiranja dubine.

Testovi 1-5 ilustrovani su na slikama 6.19 i 6.20. Disjunktnost opsega x i opsega y koordinata potrebna za izvođenje prva dva testa se veoma jednostvno proverava. Da bismo proverili da li se ceo poligon nalazi sa iste, odnosno suprotne strane drugog poligona u odnosu na posmatrača, dovoljno je da proverimo položaj svih temena poligona u odnosu na ravan kojoj pripada drugi poligon. Poslednji test je najsloženiji i podrazumeva ispitivanje da li postoji prava koja razdvaja ova dva poligona; ako ona postoji onda ona sadrži stranicu poligona jednog od poligona, te treba proći skupom svih stranica oba poligona i proveriti da li ovaj uslov važi.

Ako za svaki od poligona Q bar jedan od testova uspe, onda se poligon P iscrtava algoritmom rasterizacije i naredni poligon u datom poretku dobija ulogu poligona P. Ako za poligone P i Q nijedan od ovih testova ne uspe, onda smatramo da se poligoni P i Q potencijalno preklapaju, te proveravamo da li se možda poligon Q može iscrtati pre poligona P. Testove 1, 2 i 5 ne treba ponavljati jer su u njima uloge poligona P i Q simetrične, a testove 3 i 4 treba ponoviti sa zamenjenim ulogama poligona P i Q:

3′. da li je čitav poligon Q sa suprotne strane ravni poligona P u odnosu na posmatrača?

4′. da li je čitav poligon P sa iste strane ravni poligona Q kao i posmatrač?

Ako jedan od ovih testova uspe, onda poligoni Q i P zamenjuju uloge.

Primer 6.1. U primeru (a) sa slike 6.18 uspeva test 3′, pa se poligon Q pomera na kraj liste i uzima ulogu novog poligona P. U primeru (b) ne uspeva nijedan od testova $1-5$ niti testovi $3'$ i $4'$, te je potrebno ili poligon P ili poligon Q podeliti na delove u odnosu na ravan koja sadrži drugi polazni poligon. Primer (c) sa slike 6.18 ilustruje mogućnost beskonačne petlje u algoritmu; da bi ona bila izbegnuta, vrši se označavanje svakog poligona koji se pomera na kraj liste. Na taj način kad god nijedan od prvih pet testova ne uspe i tekući poligon Q je već obeležen, ne primenjujemo iznova testove 3′ i 4′ već delimo poligon P ili poligon Q i u listi poligona ih zamenjujemo njihovim delovima.

6.5 Pitanja

6.1 Koji je cilj konzervativnih algoritama za utvrđivanje vidljivosti?

6.2 Navesti tri osnovna algoritma konzervativnog određivanja vidljivosti.

6.3 Na koji način je moguće utvrditi da li je neki poligon sa prednje ili zadnje strane objekta?

6.4 Šta je primarna vidljivost? Zašto je važno definisati pojam vidljivosti za proizvoljan par tačaka, a ne samo za poziciju kamere i tačku sa scene?

6.5 Šta je upit preseka?

6.6 Iz kojih se koraka sastoji rej kasting algoritam?

6.7 Koja je razlika između algoritama prostora slika i algoritama prostora objekata?

6.8 Šta je bafer dubine? Koje su njegove moguće primene?

6.9 Kako funkcioniše z-bafer algoritam?

6.10 Kada se u z-bafer algoritmu tekuća vrednost u baferu menja novom vrednošću?

6.11 Da li je potrebno sortiranje poligona u z-bafer algoritmu?

6.12 Koja je ideja algoritama sa listama prioriteta?

6.13 Šta je ključ sortiranja u algoritmu sortiranja dubine?

6.14 Iz kojih koraka se sastoji algoritam sortiranja dubine?

6.15 Šta se preduzima u algoritmu sortiranja dubine kada za dva poligona ne uspeva nijedan od testova?

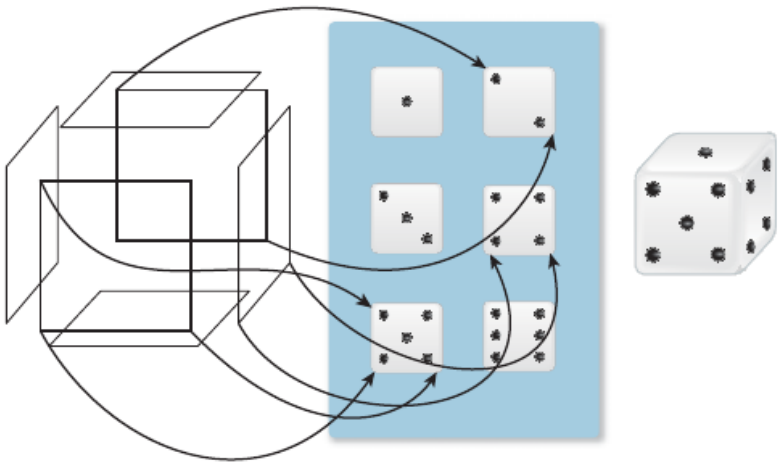
Glava 7

Teksture

Tehnika *preslikavanja (mapiranja) tekstura* (eng. texture mapping) predstavlja postupak kojim se pikseli sa *slike (mape) teksture* preslikavaju na trodimenzionalnu površ kako bi se popravio kvalitet računarski generisane slike. Preslikavanjem tekstura moguće je povećati nivo detaljnosti scene bez uvećanja broja iscrtanih primitiva. Takođe, tokom procesa renderovanja mogu se samoj površi dodati fini detalji čime se dodatno unapređuje realizam na slici: moguće je prikazati neravnine, prljavštinu, ogrebotine, otiske i dr. Dodatno, nanošenjem tekstura na površ može se postići utisak da je objekat koji se renderuje napravljen od nekog materijala, kao što je drvo, staklo ili metal.

Osnovni element teksture naziva se *teksel* (eng. texel). Kao što se rasterske slike predstavljaju nizovima piksela, teksture se predstavljaju kao nizovi teksela. Videćemo na primeru kasnije da jedan teksel sa teksture ne mora nužno da odgovara jednom pikselu iscrtane slike.

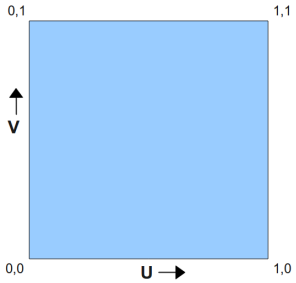
Teksture su potrebne i u situacijama kada želimo da prikažemo grube materijale ili materijale čija boja varira (npr. cigla, šljunak, mermer ili drvo) ili da napravimo pozadinu poput travnate ravnice ili guste šume. Naime, ni u ovim situacijama se ne preporučuje pravljenje mreže poligona koja bi sadržala sve detalje fino graduisane strukture materijala jer bi dobijeni model bio jako složen. Razmotrimo složenost mreže kojom bi se modelovale pukotine i rupice grubo klesanog kamena antičkih piramida – od jednostavnog modela piramide u vidu mreže od četiri trougla dobili bismo milione trouglova čime bi eksplodirali i memorijski i vremenski zahtevi obrade. Naime, iako grafičke kartice mogu da renderuju više miliona poligona u sekundi, to nije praktično izvodljivo za aplikacije koje rade u realnom vremenu. I u ovoj situaciji se preslikavanjem tekstura mogu simulirati složene scene i složeni materijali bez povećanja složenosti same mreže trouglova. Tehniku preslikavanja tekstura predložio je Edvin Katmul 1974. godine.



Slika 7.1: Temenima svake od šest strana kocke dodeljene su teksturne koordinate (neke od njih prikazane su strelicama). Jedno isto teme ima tri pridružene različite teksturne koordinate jer je deo tri strane kocke.

Teksture preslikavamo na poligon (ili skup poligona) tako što svakoj tački poligona dodeljujemo boju čitanjem vrednosti sa slike teksture (slika 7.1). Nanošenje tekstura na trodimenzionu površ nalikuje

prekrivanju objekta rastegljivim listom papira. Preciznije, za svaku tačku P date površi potrebno je zadati koju tačku papira treba da dotakne. U praksi se ovo preslikavanje zadaje za svako teme površi, dok se za unutrašnje tačke površi koristi interpolacija. Ovakav mehanizam nanošenja tekstura podrazumeva postojanje koordinatnog sistema u kom možemo da referišemo na pozicije sa slike teksture. Prema dogovoru na tačke sa slike teksture referišemo korišćenjem *teksturnih koordinata* (eng. texture coordinates) u i v koje uzimaju vrednosti iz intervala $[0, 1]$ (slika 7.2). Teksturne koordinate temena se unapred izračunavaju i zadaju za svako teme. Primetimo da isto teme objekta može imati različite teksturne koordinate za različite strane objekta kojima ono pripada (slika 7.1).



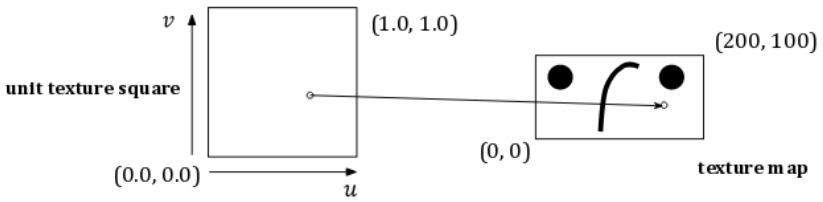
Slika 7.2: Koordinatni sistem teksture.

7.1 Preslikavanje tekstura

Preslikavanje tekstura se sastoji od dve komponente: prva komponenta predstavlja *parametrizaciju površi*, odnosno preslikavanje trodimenzione površi u ravan. Formalno, potrebno je odrediti funkciju $f(x, y, z) = (u, v)$ koja opisuje površ. Dakle, korišćenjem parametrizacije površi svakoj tački površi pridružuju se odgovarajuće teksturne koordinate u i v. Druga komponenta odgovara samom postupku *teksturisanja*, odnosno dodeljivanju vrednosti teksture svakoj tački iz (u, v) ravni, na osnovu njenih koordinata. Pritom vrednosti koje se dodeljuju mogu biti boje, ali i mnogi drugi vizualni parametri, poput vektora normala površi, eksponenta spekularne refleksije i slično.

7.1.1 Teksturisanje

Razmotrimo najpre fazu teksturisanja, kao jednostavniju komponentu procesa preslikavanja tekstura. Razlikujemo dve vrste teksturisanja: *teksturisanje na osnovu slike*, kada se koordinate (u, v) koriste za indeksiranje pozicije na slici koja sadrži teksturu i *proceduralno teksturisanje*, kada su koordinate (u, v) ulaz u neku funkciju kojom se računa vrednost teksture. Teksturisanje na osnovu slike je češće u upotrebi. Kod njega se u opštem slučaju vrši preslikavanje tačke (u, v) jediničnog kvadrata na teksturu širine w i visine h, tako što se tačka sa teksturnim koordinatama $(0, 0)$ slika u donji levi ugao mape teksture, tačka sa teksturnim koordinatama $(1, 1)$ slika u gornji desni ugao mape teksture, a tačka sa koordinatama (u, v) u tačku sa slike sa koordinatama $(u \cdot w, v \cdot h)$ (slika 7.3). Primetimo da se pri ovom računanju mogu dobiti necelobrojne koordinate tačke sa slike teksture, pa je potrebno vratiti ili vrednost u najbližoj celobrojnoj tački ili izvršiti uprosečavanje vrednosti susednih elemenata slike teksture.



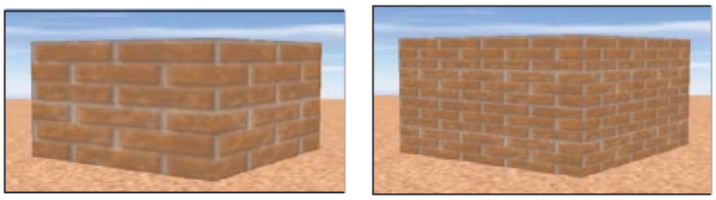
Slika 7.3: Ilustracija postupka teksturisanja na osnovu slike.

Tehnike nanošenja tekstura

Postoje različiti stilovi preslikavanja tekstura. Ako se tekstura koristi da bi oponašala materijal koji je konzistentnog izgleda i bez očiglednih tačaka prekida, kao što je pesak, asfalt ili cigla, ciljna površ se

može pokriti ponavljanjem slike. U ovom slučaju tekstura je obično mala slika materijala (dobijena ili sintetički ili fotoaparatom), dizajnirana tako da se pri popločavanju susedne "pločice" neprimetno uklapaju. Na primer, razmotrimo sliku teksture sa slike 7.4 koja prikazuje šest redova cigala. Ako je preslikamo na svaku stranu kvadra bez ponavljanja dobićemo realističnu sliku neke niže građevine (slika 7.5 a)). Međutim, na ovaj način ne bi mogla da se predstavi građevina koja se sastoji od velikog broja redova cigala. *Popločavanje* (eng. tiling) je tehnika nanošenja teksture na poligon tako što se veći broj kopija teksture postavlja jedna do druge, stvarajući pritom privid jedinstvene slike, bez uočavanja mesta gde se spajaju dve slike teksture. Ona omogućava da se broj redova cigala umnoži čime se može dobiti slika visokog utvrđenja koje se sastoji iz npr. duplo većeg broja redova cigala (slika 7.5 b)).

Slika 7.4: Kvadratna slika uzorka cigala.



Slika 7.5: a) Rezultat istezanja jedne kopije teksture cigala na svaki od zidova. b) Rezultat popločavanja višestrukim kopijama teksture cigle na svaki od zidova.

Nekada se tekstura koristi kao zamena za jako složeni model, npr. grad koji se gleda sa visine ili oblačno nebo. Tada je slika teksture po pravilu visoke rezolucije i ona može nastati fotoaparatom ili sintetički (ako npr. predstavlja reljef u nekom svetu fantazije). Ako bismo izvršili popločavanje ovom vrstom teksture, na slici bi se mogli uočiti obrasci koji se pravilno ponavljaju i scena bi dobila neprirodan izgled. Stoga se kod ovakve vrste tekstura koristi tehnika *istezanja* (eng. stretching), tj. teksturne koordinate se postavljaju tako da slika teksture pokrije celu mrežu. Na primer, za prikaz neba mogla bi se iskoristiti slika teksture data na slici 7.6 koja bi se istezala na unutrašnjoj strani valjka (slika 7.7).

Slika 7.6: Slika neba koja se može iskorisiti kao slika teksture.

Slika 7.7: Prikaz neba dobijen skaliranjem slike neba kao teksture.

7.1.2 Parametrizacija površi

Preslikavanje dvodimenzione teksture na površ u trodimenzionom prostoru zahteva parametrizaciju površi u terminima teksturnih koordinata u i v. Zadatak parametrizacije površi jeste određivanje funkcije f koordinata x, y i z kojom se opisuje površ: $f(x, y, z) = (u, v)$. Zadatak parametrizacije proizvoljne površi je težak problem. Mi ćemo u ovom materijalu razmotriti parametrizaciju geometrijskih primitiva koje se lako parametrizuju i parametrizaciju proizvoljne mreže trouglova.

Parametrizacija osnovnih geometrijskih tela

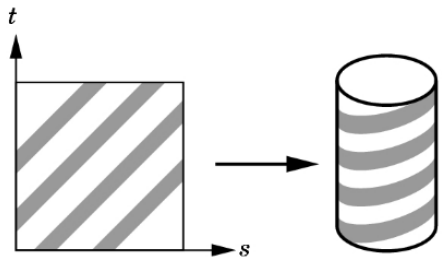
Neke trodimenzione primitive, kao što su valjak, kupa i sfera se prirodno definišu kao parametarske funkcije, tj. nije teško odrediti funkciju g tako da važi: $g(u, v) = (x, y, z)$. Razmotrimo preslikavanje teksturnih koordinata u tačku sa date površi. Potrebno je odrediti funkcije:

$$\begin{aligned} x &= x(u, v) \\ y &= y(u, v) \\ z &= z(u, v) \end{aligned}$$

kojima se vrši preslikavanje jediničnog kvadrata parametrizovanog koordinatama u i v u tačku date površi.

Preslikavanje pravougaonika parametrizovanog sa u i v na valjak poluprečnika r i visine h čiji je centar osnove u koordinatnom početku može se zadati na naredni način (slika 7.8):

$$\begin{aligned} x &= r \cdot \cos 2\pi u \\ y &= v/h \\ z &= r \cdot \sin 2\pi u \end{aligned} \tag{7.1}$$



Slika 7.8: Ilustracija preslikavanja pravougaone mape na valjak.

Uglom $\theta = 2\pi u$ zadat je ugao koji projekcija tačke P sa koordinatama x, y i z na ravan Oxz zaklapa sa x osom. Koordinatama x i z se definiše ugao koji tačka P zahvata sa osom valjka, dok se koordinatom y zadaje visina tačke P.

Analogno, možemo izvršiti preslikavanje pravougaonika parametrizovanog sa u i v na sferu poluprečnika r sa centrom u koordinatnom početku (slika 7.9):

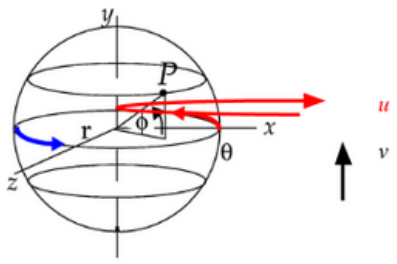
$$\begin{aligned} x &= r \cdot \cos 2\pi u \\ y &= r \cdot \sin 2\pi u \cdot \cos 2\pi v \\ z &= r \cdot \sin 2\pi u \cdot \sin 2\pi v \end{aligned} \tag{7.2}$$

Sa $\phi = 2\pi u$ zadat je ugao koji tačka P zaklapa sa svojom projekcijom na ravan Oxz, a sa $\theta = 2\pi v$ ugao koji ta projekcija tačke P zaklapa sa x osom. U slučaju preslikavanja pravougaone mape na sferu javlja se distorzija na polovima.

Međutim, nama je u procesu rasterizacije potrebno obratno: da za datu tačku P sa koordinatama x, y, z objekta odredimo kojoj tački sa teksture odgovara. Dakle, potrebno je odrediti preslikavanje oblika:

$$\begin{aligned} u &= u(x, y, z) \\ v &= v(x, y, z) \end{aligned}$$

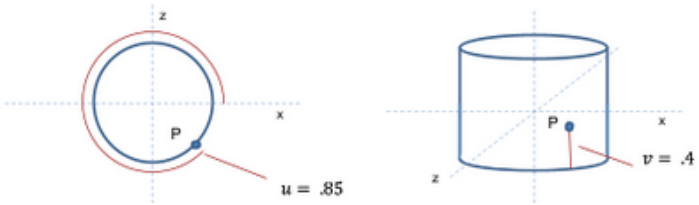
Razmotrimo rešavanje ovog problema kada je data tačka P koja pripada površi koja je oblika jediničnog valjka sa osnovom i ravni Oxz čije je središte u koordinatnom početku. Ako se tačka P nalazi na nekoj



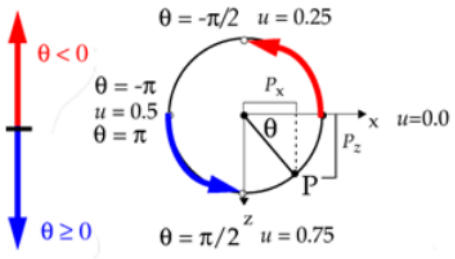
Slika 7.9: Računanje u i v koordinate u slučaju sfere.

od osnova valjka, izvršićemo preslikavanje kao da je u pitanju ravan. Ukoliko se nalazi na omotaču valjka, onda ćemo iskoristiti poziciju tačke u odnosu na obim kruga da bismo izračunali u koordinatu, a visinu tačke iskoristićemo za računanje koordinate v (slika 7.10). Koordinatu v dobijamo direktno: preslikavanjem opsega $[-0.5, 0.5]$ za jedinični valjak u opseg $[0.0, 1.0]$ dodavanjem vrednosti 0.5. Koordinatu u računamo na osnovu ugla θ koji projekcija tačke P na ravan Oxz zahvata sa pozitivnim delom x ose, računanjem vrednosti $\arctan\theta$. Međutim, pošto je kodomen funkcije $\arctan$ jednak $(-\frac{\pi}{2}, \frac{\pi}{2})$, dve tačke sa obima daju istu vrednost ugla θ, tako da moramo razmatrati dva slučaja (slika 7.11).

$$\begin{aligned} \theta &= \arctan\frac{z}{x} \\ u &= \begin{cases} \frac{\theta}{2\pi} \text{ za } \theta < 0 \\ 1 - \frac{\theta}{2\pi} \text{ za } \theta \geq 0 \end{cases} \\ v &= y + 1/2 \end{aligned}$$



Slika 7.10: Računanje u i v koordinate u slučaju valjka.



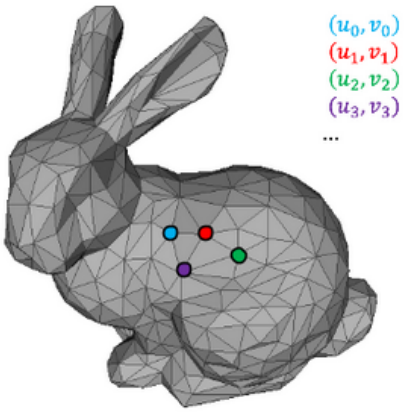
Slika 7.11: Računanje u i v koordinate u slučaju valjka.

Kada je u pitanju sfera, koordinatu u računamo na isti način kao i u slučaju valjka, dok se koordinata v dobija kao funkcija geografske širine ϕ (slika 7.9). Jedino se na polovima (za $v = 0$ i $v = 1$) javljaju singularnosti i tada se vrednost u postavlja na neku predefinisanu vrednost, npr. $u = 0.5$.

$$\begin{aligned} \theta &= \arctan\frac{z}{x} \\ u &= \begin{cases} \frac{\theta}{2\pi} \text{ za } \theta < 0 \\ 1 - \frac{\theta}{2\pi} \text{ za } \theta \geq 0 \end{cases} \\ \phi &= \arcsin\frac{y}{r} \\ v &= \frac{\phi}{\pi} + \frac{1}{2} \end{aligned}$$

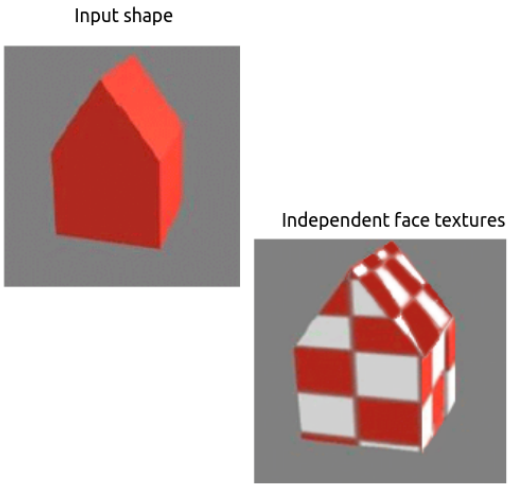
Parametrizacija mreže trouglova

Razmotrimo situaciju kada je potrebno teksturisati proizvoljnu mrežu trouglova. Najpre je potrebno parametrizovati ovu mrežu, tj. svakoj tački sa mreže trouglova pridružiti teksturne koordinate u i v. Ovakva preslikavanja u opštem slučaju nije lako odrediti jer ne postoji funkcija kojom se može definisati oblik proizvoljne mreže trouglova. Umesto toga, svakom temenu mreže pridružićemo odgovarajuće (u, v) koordinate, a nakon toga se jednostavno može odrediti (u, v) koordinate za svaku tačku date mreže (slika 7.12).



Slika 7.12: Definisanje (u, v) koordinata za svako teme mreže trouglova (preuzeto sa slajdova Univerziteta Brown).

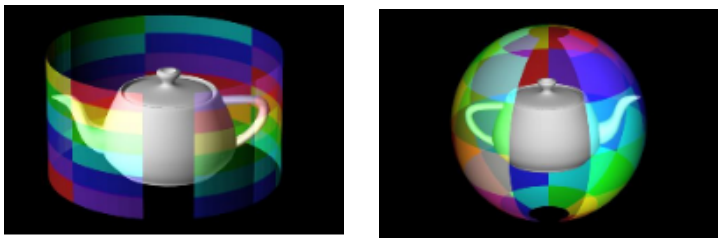
Najjednostavniji vid preslikavanja je taj kod koga se svaka poligonalna strana mreže nezavisno teksturiše. Međutim, ovakav način teksturisanja može iznedriti ružne spojeve na mestima gde se strane mreže sustiču.



Slika 7.13: Ilustracija nezavisnog teksturisanja svake strane mreže.

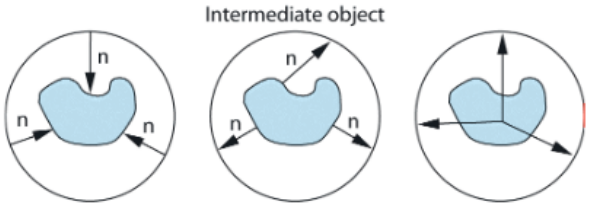
Alternativno, može se izvršiti tzv. *dvostepeno preslikavanje* (eng. two-stage mapping) kojim se tekstura najpre preslikava na neku jednostavnu međupovrš koja je kanonskog oblika (npr. sferu ili valjak), a nakon toga se vrši preslikavanje sa te međupovrši na ciljnu površ. Ovo daje dobre rezultate ukoliko je oblik koji treba parametrizovati približno sferni, odnosno cilindrični. Dakle, najpre se oko objekta opisuje granični opseg (slika 7.14). Nakon toga se preslikavanje izvodi u dva koraka: prvo se vrši preslikavanje teksture na taj granični opseg, a zatim preslikavanje sa graničnog opsega na ciljni objekat. Pritom je drugo pomenuto preslikavanje moguće uraditi na više različitih načina (slika 7.15):

- određivanjem preseka normale graničnog opsega sa ciljnom površi,
- određivanjem preseka normale ciljne površi sa graničnim opsegom,



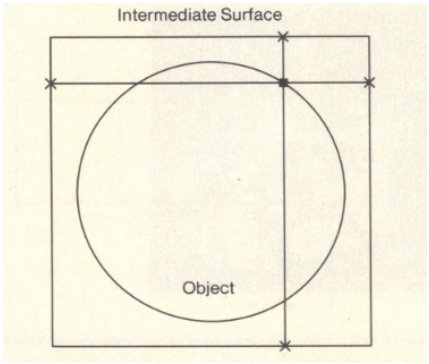
Slika 7.14: Opisivanje graničnog opsega u obliku valjka (levo) i sfere (desno) oko ciljne površi.

- određivanjem preseka vektora iz centra ciljne površi sa graničnim opsegom objekta.



Slika 7.15: Različiti načini preslikavanja sa graničnog opsega na ciljnu površ.

Ukoliko se preslikavanje sa graničnog opsega na ciljnu površ vrši računanjem preseka normale graničnog opsega sa ciljnom površi, može se desiti da se veći broj različitih tačaka sa graničnog opsega preslikava u jednu istu tačku sa ciljne površi (slika 7.16). Ovaj problem se može rešiti odabirom tačke sa graničnog opsega koja je najbliža ciljnoj površi u smeru date normale.



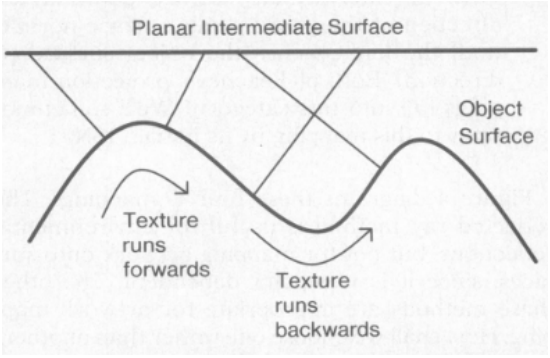
Slika 7.16: Ilustracija problema nejednoznačnog preslikavanja u ravni. Četiri tačke sa graničnog opsega oblika kvadrata preslikavaju se u istu tačku kruga. (preuzeto sa http://www1.cs.columbia.edu/~allen/PHOTOPAPERS/biersloantexma.pdf).

Primetimo da ako se računa presek normale ciljne površi i graničnog opsega i ako ciljna površ nije konveksna, onda se tekstura prikazuje u suprotnom smeru u oblastima gde je površ konkavna (slika 7.17). Naime, ako se krećemo duž neke linije, tekstura može biti prikazana u jednom smeru, pa u njemu suprotnom, pa nakon toga opet u početnom smeru, itd.

Međutim, u mnogim primenama, poput video igara ili filmova, objekti na koje je potrebno preslikati teksture su znatno komplikovaniji od onih koje smo ovde razmatrali. U tim situacijama je potrebno imati visok nivo kontrole nad tim kako će tekstura izgledati na samom objektu (npr. u slučaju teksture ljudskog lica koje je potrebno preslikati na geometriju odgovarajućeg modela), pa se vrši namensko modelovanje tekstura u odgovarajućim grafičkim programima.

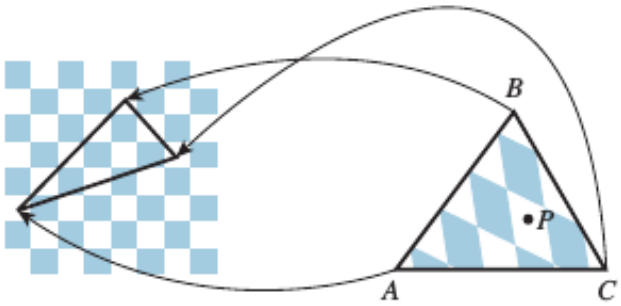
7.1.3 Interpolacija vrednosti u trouglu

Pretpostavimo da imamo mrežu trouglova u kojoj su svakom temenu trougla dodeljene teksturne koordinate. Postavlja se pitanje kako odrediti teksturne koordinate neke tačke unutar trougla. Dodela vrednosti svakom temenu trougla nekog svojstva koje se linearno menja duž trougla na jedinstven način



Slika 7.17: Ako se preslikavanje sa graničnog opsega na ciljnu površ radi određivanjem preseka normale ciljne površi sa graničnim opsegom, konkavnost površi uzrokuje da se uzorak sa teksture prikaže u obratnom smeru (preuzeto sa http://www1.cs.columbia.edu/~allen/PHOTOPAPERS/biersloantexma.pdf).

određuje vrednost svake tačke unutar trougla: ako je P tačka trougla ABC i u koordinate tačaka A, B i C su redom u_A, u_B i u_C, u koordinatu tačke P možemo odrediti korišćenjem *baricentričnih koordinata* (eng. barycentric coordinates). Ovaj postupak ilustrovan je na slici 7.18.



Slika 7.18: Boja tačke P trougla ABC određuje se na osnovu mape teksture. Temenima A, B i C trougla pridružene su odgovarajuće tačke na mapi teksture koja predstavlja šahovsku tablu, dok tački P odgovara tačka sa belog polja šahovske table, te će njena boja u trouglu ABC biti bela.

Razmotrimo najpre postupak interpolacije između dve date tačke A_1 i A_2: svaku tačku Q sa duži A_1A_2 možemo izraziti kao:

$$Q = P(t) = tA_1 + (1-t)A_2$$

gde je $0 \leq t \leq 1$. Ovu jednačinu možemo drugačije zapisati kao:

$$Q = P(t_1, t_2) = t_1A_1 + t_2A_2$$

pri čemu važi $t_1 + t_2 = 1$ i $t_1, t_2 \geq 0$. Vrednosti t_1 i t_2 nazivamo baricentričnim koordinatama duži određene tačkama A_1 i A_2.

Vrednost proizvoljnog svojstva koje se linearno menja duž duži A_1A_2 može se izraziti kao $f(Q) = t_1 \cdot f(A_1) + t_2 \cdot f(A_2),\ t_1 + t_2 = 1,\ t_1, t_2 \geq 0$.

Baricentrične koordinate možemo uopštiti na trougao. Naime, ako trougao ima temena A, B i C, tada tačka $Q = (1-t)A + tB$, za $0 \leq t \leq 1$ predstavlja proizvoljnu tačku stranice AB trougla ABC. Slično, tačka oblika $P = (1-s)Q + sC$ za $0 \leq s \leq 1$ predstavlja proizvoljnu tačku duži QC. Za proizvoljnu tačku Q stranice AB, tačka P odgovaraće proizvoljnoj tački trougla ABC. Zamenom jednačine tačke Q u jednačinu tačke P dobijamo:

$$P = (1-s)(1-t)A + (1-s)tB + sC$$

Na ovaj način definiše se funkcija dve promenljive s i t:

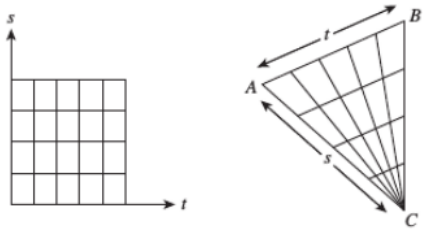
$$F : [0,1] \times [0,1] \to R^2 : (s,t) \to (1-s)(1-t)A + (1-s)tB + sC$$

čija je slika tačno trougao ABC (slika 7.19). Lako se pokazuje da su za $0 \leq s, t \leq 1$ sva tri koeficijenta $(1-s)(1-t)$, $(1-s)t$ i s nenegativna. Dodatno, njihovim sumiranjem dobijamo:

$$(1-s)(1-t) + (1-s)t + s = (1-s) + s = 1$$

Dakle, proizvoljnu tačku trougla ABC možemo predstaviti kao konveksnu linearnu kombinaciju temena trougla, odnosno u obliku $\alpha A + \beta B + \gamma C$, gde je $\alpha + \beta + \gamma = 1$ i $\alpha, \beta, \gamma \geq 0$. Tačke kod kojih važi $\alpha = 0$ leže na stranici BC, tačke kod kojih važi $\beta = 0$ leže na stranici AC, dok tačke kod kojih važi $\gamma = 0$ leže na stranici AB. Brojeve α, β, γ nazivamo baricentričnim koordinatama tačke P u odnosu na trougao ABC i pišemo $P = \alpha A + \beta B + \gamma C$.

Dakle, za teksturnu koordinatu u tačke P važi $u_P = \alpha u_A + \beta u_B + \gamma u_C$. Analogno možemo izraziti i v koordinatu tačke P preko koordinata v temena A, B i C. Na ovaj način se na jedinstven način određuju teksturne koordinate tačke P, interpolacijom teksturnih koordinata temena trougla ABC.

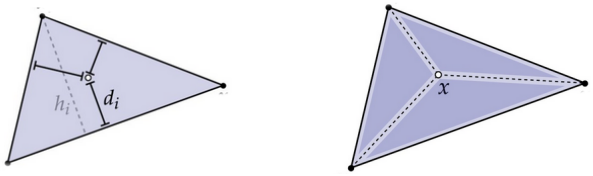


Slika 7.19: Funkcija F slika jedinični kvadrat u trougao ABC; ovom funkcijom se ivica kvadrata za $s = 1$ slika u tačku C. Sve ostale duži jediničnog kvadrata sa slike slikaju se u odgovarajuće duži unutar trougla ABC.

Baricentrične koordinate tačke P možemo da tumačimo na različite načine: one odgovaraju rastojanjima tačke P od stranica trougla. Naime, težina uz teme trougla treba da bude proporcionalna rastojanju tačke P od naspramne stranice trougla: npr. koordinata uz teme A_1 treba da odgovara rastojanju tačke P do stranice A_2A_3. Preciznije, baricentrične koordinate tačke P možemo dobiti tako što rastojanja tačke P do stranica A_1A_2, A_2A_3 i A_3A_1 podelimo redom dužinama visina trougla $\triangle A_1A_2A_3$ iz temena A_3, A_1 i A_2 (slika 7.20 levo):

$$t_i = \frac{d_i}{h_i}$$

Baricentrične koordinate tačke P možemo da tumačimo i preko površina trouglova: težina uz teme trougla treba da bude proporcionalna površini trougla koji tačka P gradi sa druga dva temena trougla (slika 7.20 desno).



Slika 7.20: Tumačenje baricentričnih koordinata putem rastojanja do naspramnih stranica trougla i putem odnosa površina trouglova.

Preciznije, baricentrične koordinate tačke P dobijamo tako što površine trouglova $\triangle PA_1A_2$, $\triangle PA_2A_3$ i $\triangle PA_3A_1$ podelimo površinom $\triangle A_1A_2A_3$, redom. Na primer:

$$t_1 = \frac{p(\triangle PA_2A_3)}{p(\triangle A_1A_2A_3)}$$

7.1.4 Perspektivno nekorektna interpolacija

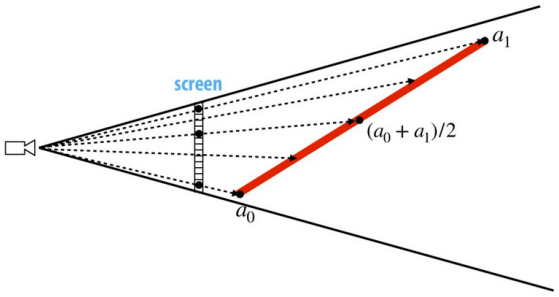
Razmotrimo problem interpolacije neke linearne funkcije f nad trouglom ABC u trodimenzionom prostoru, čije su vrednosti zadate u tačkama A, B i C. Označimo sa $A'B'C'$ projekciju trougla ABC na ravan slike. Kako bismo znali kojom bojom treba obojiti neki piksel P u projekciji trougla, računamo baricentrične koordinate α, β i γ tačke P u odnosu na trougao $A'B'C'$:

$$P = \alpha A + \beta B + \gamma C$$

Ukoliko bismo vrednost $f(P)$ računali kao

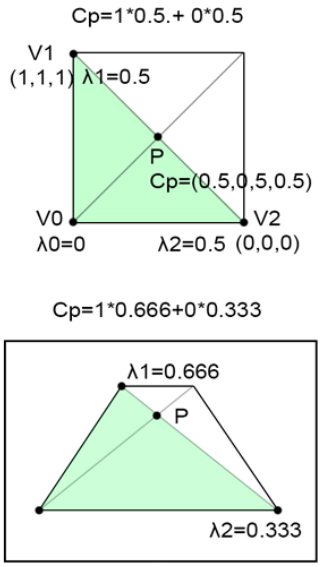
$$f(P) = \alpha f(A) + \beta f(B) + \gamma f(C)$$

, ne bismo dobili ispravnu vrednost. Naime, zbog perspektivne projekcije, interpolacija vrednosti na trouglu sa različitim dubinama ne daje dobre rezultate. Naime, perspektivna projekcija nije afina funkcija koordinata ekrana, pa se odnosi rastojanja između kolinearnih tačaka ne čuvaju. Specijalno, tačka koja je u središtu duži u trodimenzionom prostoru, ne mora se slikati u središte projektovane duži u ravni ekrana (slika 7.21).



Slika 7.21: Središte duži se pri perspektivnoj projekciji ne slika u središte duži.

Ilustrujmo ovaj problem na još jednom primeru. Neka je dat kvadrat u trodimenzionom prostoru koji nije paralelan ravni projekcije i neka se on projektuje u paralelogram (slika 7.22). Razmotrimo položaj tačke P u trodimenzionom prostoru objekta i u dvodimenzionom prostoru projekcije: naime, za tačku P kao presek dijagonala kvadrata važi: $P = 0.5 \cdot V_1 + 0.5 \cdot V_2$, dok za tačku P' kao presek dijagonala paralelograma važi: $P' = 0.666 \cdot V_1' + 0.333 \cdot V_2'$. Dakle, središte duži V_1V_2 se ne slika u središte projekcije te duži, te ne možemo koristiti baricentrične koordinate u odnosu na projektovani trougao za interpolaciju vrednosti u trodimenzionom prostoru. Naime, potrebno je vrednosti funkcije f interpolirati linearno u trodimenzionom prostoru objekta, a ne u prostoru slike.



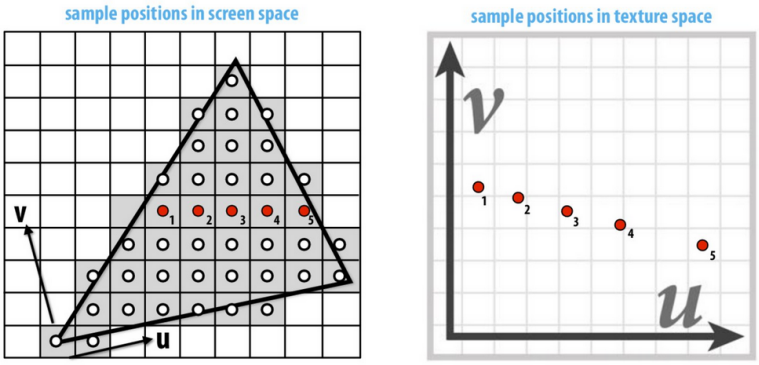
Slika 7.22: Ilustracija kvadrata čije se središte slika u središte paralelograma.

Ovaj problem se može rešiti na sledeći način: najpre se izračunava z koordinata svakog temena, a zatim i vrednosti $Z = 1/z$ i $g = f/z$ u svakom od temena. Nakon toga vršimo interpolaciju vrednosti Z i g korišćenjem baricentričnih koordinata projekcije trougla. Konačno, za svaki od fragmenata potrebno je interpolisanu vrednost funkcije g podeliti interpolisanom vrednošću funkcije Z kako bismo dobili finalnu vrednost funkcije f fragmenta[1].

[1]Izvođenje ovog tvrđenja dostupno je na adresi: https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation/perspective-correct-interpolation-vertex-attributes.html

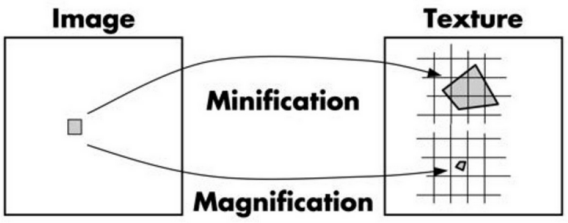
7.2 Uzorkovanje teksture

Razmotrimo teksturisanje proizvoljne mreže trouglova. Pikseli u prostoru ekrana odgovaraće regionima promenljive veličine i lokacije na teksturi. Naime, lokacije u kojima vršimo uzorkovanje imaju ravnomernu raspodelu u prostoru ekrana, dok to nužno ne važi za odgovarajuće lokacije u prostoru tekstura (slika 7.23). Nepravilan obrazac uzorkovanja dovodi do efekta poznatog kao *teksturni alijasing*.



Slika 7.23: Razlika u raspodeli tačaka u kojima se vrši uzorkovanje u prostoru ekrana i u prostoru tekstura.

Prilikom teksturisanja razlikujemo dva granična scenarija: prvi se odnosi na situaciju kada je kamera jako blizu objekta, a drugi kada je objekat jako daleko od kamere. Naime, u prvoj razmatranoj situaciji, s obzirom da je objekat blizu kamere, pojedinačni piksel ekrana se preslikava u veoma mali region teksture i tada dolazi do *uvećanja (magnifikacije) teksture* (eng. texture magnification). U drugom scenariju objekat je daleko od kamere, te se pojedinačni piksel ekrana preslikava u veliki region teksture i tu pojavu nazivamo *umanjenje (minifikacija) teksture* (eng. texture minification) (slika 7.24).

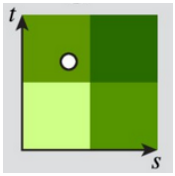


Slika 7.24: Uvećanje vs. umanjenje teksture.

7.2.1 Uvećanje teksture

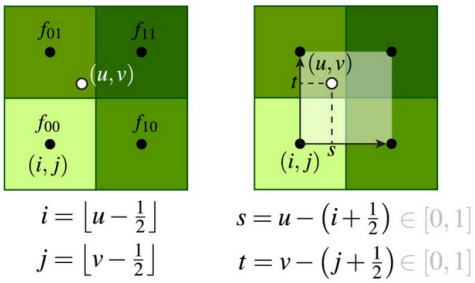
Kada se objekat posmatra sa veoma male udaljenosti čak ni mapa teksture ne može da sadrži dovoljno detalja: tada je moguće da se dva susedna teksela preslikavaju u dva piksela finalne slike koji mogu biti desetinu piksela daleko. Dakle, piksel ekrana se u ovom slučaju preslikava u veoma mali region teksture. Pritom se može desiti da tačka teksture u kojoj treba pročitati vrednost ima necelobrojne vrednosti koordinata, pa se postavlja pitanje kako dobiti vrednost teksture na necelobrojnoj lokaciji. Najjednostavniji način je uzeti vrednost najbližeg elementa (teksela) teksture – ovu metodu nazivamo *metoda najbližeg suseda* (slika 7.25). Ova metoda je veoma efikasna, međutim, daje sliku sačinjenu od kvadratića uniformne boje. Ako bismo razmatrali situaciju u kojoj se kamera prilažava objektu, taj efekat bi bio sve uočljiviji.

Drugi, nešto složeniji pristup, koji daje lepši vizuelni prikaz je *bilinearna interpolacija* (eng. bilinear interpolation), kod koje vršimo interpolaciju vrednosti najpre u horizontalnom smeru, a zatim u vertikalnom. To postižemo računanjem težinskog proseka četiri najbliža teksela (slika 7.26), pri čemu težinski koeficijenti odgovaraju rastojanju do tih teksela (slika 7.27). Naime, ako računamo vrednost u tački (x, y), a koordinate susedna četiri teksela su (x_i, y_i), (x_{i+1}, y_i), (x_i, y_{i+1}) i (x_{i+1}, y_{i+1}), i važi $x = x_i + x_f$ i $y = y_i + y_f$, onda vrednost u tački (x, y) računamo tako što najpre vršimo interpolaciju po koordinati x, a zatim po koordinati y:

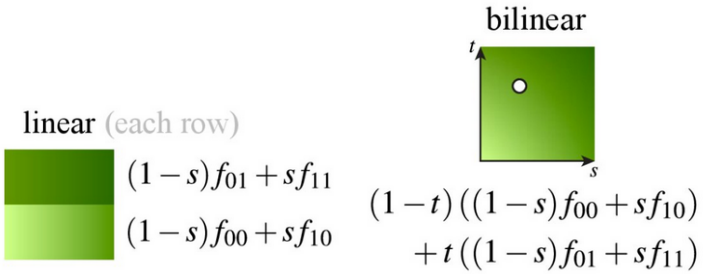


Slika 7.25: Ilustracija metode najbližeg suseda.

$$
\begin{aligned}
I(x,y) &= (1-x_f)I(x_i,y) + x_f I(x_{i+1},y) \\
&= (1-x_f)(I(x_i,y_i)(1-y_f) + I(x_i,y_{i+1})y_f) \\
&+ x_f(I(x_{i+1},y_i)(1-y_f) + I(x_{i+1},y_{i+1})y_f) \\
&= I(x_i,y_i)(1-x_f)(1-y_f) + I(x_{i+1},y_i)x_f(1-y_f) \\
&+ I(x_i,y_{i+1})(1-x_f)y_f + I(x_{i+1},y_{i+1})x_f y_f
\end{aligned}
$$



Slika 7.26: Računanje središta susedna četiri teksela i rastojanja date tačke od tih središta.

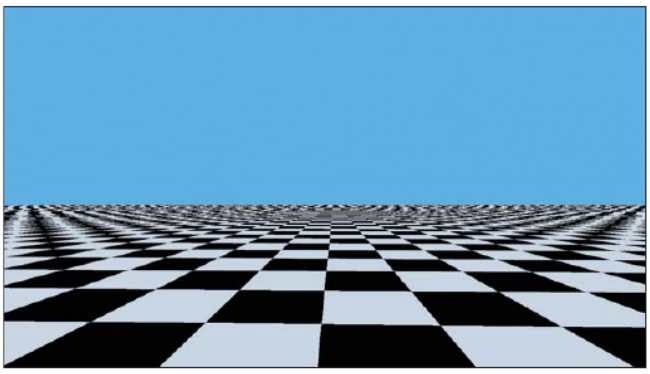


Slika 7.27: Interpolacija vrednosti u centrima četiri data teksela, najpre po horizontali, a onda po vertikali.

7.2.2 Umanjenje teksture

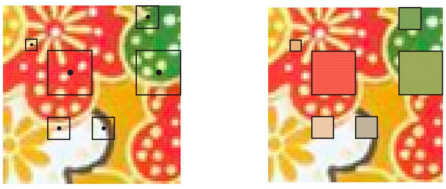
Ako trougao ima teksturne koordinate koje prekrivaju veliku površinu slike teksture, ali sam trougao prilikom renderovanja zauzima relativno mali deo finalne slike, onda svaki piksel finalne slike odgovara velikom broju teksela slike. Ovu pojavu nazivamo umanjenje teksture. Postavlja se pitanje na koji način dobiti boju piksela koja odgovara velikoj oblasti na teksturi? Ukoliko bismo samo uzeli jednu odgovarajuću vrednost sa slike teksture, koja odgovara centru regiona, javio bi se *teksturni aliasing*. Razmotrimo sliku beskonačne šahovske table koja se udaljava od posmatrača iznad koje je plavo nebo, prikazanu na slici 7.28. Boja svakog od piksela na slici je crna, bela ili plava. Možemo primeti da slika izgleda neprirodno, odnosno na horizontu se javlja tzv. *Moire uzorak* (eng. Moiré pattern).

Razmotrimo sada situaciju kada se u tom regionu nalazi veliki broj različitih boja: uzimanjem boje u centru regiona se gotovo na slučajan način bira boja. Ovo je posebno nepogodno u situaciji kada imamo kretanje, pri čemu će se centar regiona konstantno menjati, a boja piksela "skakati" sa jedne boje iz regiona na drugu. Jasno je da bi bilo idealno uzeti prosečnu vrednost regiona teksture, ali je ovo skupo izračunati (slika 7.29). Međutim, ako bismo za svaki pojedinačni piksel koji se renderuje



Slika 7.28: Teksturni aliasing koji potiče od uzorkovanja teksture šahovske table u centru regiona. Boja svakog piksela na slici je bela, crna ili plava.

na slici vršili mešanje boja odgovarajućih piksela teksturne slike, teksturisanje bi postalo jako sporo. U primeru prikaza šahovske table koja se udaljava od posmatrača, bi pored piksela crne i bele boje dobili i piksele raznih nijansi sive. Međutim, nije efikasno računati prosečnu vrednost teksture iznova za svaki od piksela slike.



Slika 7.29: Računanje vrednosti piksela kao vrednosti u središtu regiona i kao prosečne vrednosti boja iz regiona.

Jedan način da se ovaj efekat ublaži jeste *prefiltriranje* (eng. prefiltering), kod koga jednom, unapred izračunamo proseke kako bismo u vreme izvršavanja dobijene vrednosti mogli koristiti veliki broj puta. Postavlja se pitanje koje proseke treba unapred izračunati i čuvati? Jasno je da ne možemo unapred izračunati sve moguće proseke. Za objekte koji su blizu kameri potrebne su slike visoke rezolucije, ali su one neefikasne za udaljene objekte, pa bi tada bilo pogodno imati slike manje rezolucije.

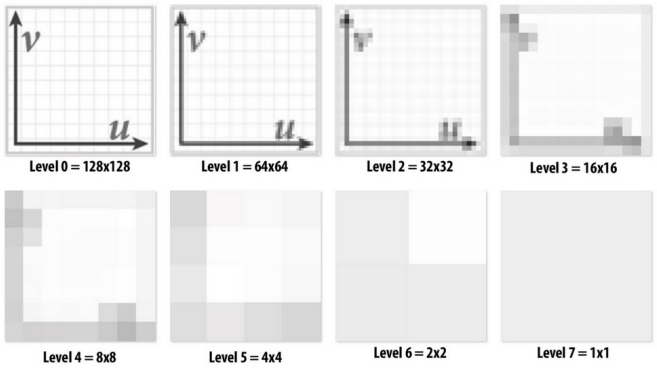
Preračunavanja unapred mogu se izvesti tehnikom *MIP preslikavanja* (eng. MIP mapping). Termin MIP predstavlja akronim latinske fraze "multum in parvo" koji znači "mnogo u malom". *Mip mape* (eng. mip map) je 1983. godine po prvi put uveo Lans Vilijams (Lance Williams) koji je predložio da se uz sliku teksture čuva i odgovarajuća piramida sačinjena od kopija te iste slike samo manjih dimenzija (slika 7.31). Na dnu piramide nalazi se originalna slika teksture, dok su visina i širina svake naredne slike duplo manje od prethodne. Na ovaj način dobijamo slike sve manje rezolucije koje sadrže prosečne vrednosti sve većih i većih regiona: tekseli na višem nivou čuvaju proseke tekstura sa prethodnog nivoa (slika 7.30). Ovako nastavljamo sve dok ne stignemo do slike dimenzije 1×1. Za udaljene objekte koristićemo teksture manje rezolucije (slika 7.31). Na ovaj način moguće je pri određivanju vrednosti nekog piskela pogledati samo jedan element MIP mape odgovarajućeg nivoa.

Mip mape se koriste za povećanje brzine renderovanja i smanjenje aliasing efekta u situacijama kada se rastojanje između objekta i kamere može promeniti. Naime, sa povećanjem udaljenosti objekta od posmatrača, tekstura objekta će se prikazivati na ekranu u manjoj rezoluciji od svoje i potrebno je odrediti boju piksela na osnovu većeg broja elemenata teksture. Tehnika mip preslikavanja može u tome pomoći tako što se slike visoke rezolucije koriste za objekte koji su bliži kameri, dok se slike manje rezolucije koriste za udaljenije objekte ili objekte manjih dimenzija.

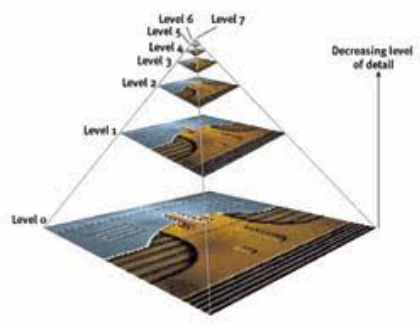
Korišćenje MIP mapa je donekle prostorno zahtevnije od čuvanja samo polazne slike teksture: MIP mape koriste 33% više memorije po teksturi jer važi:

$$\sum_{i=1}^{\infty} \frac{1}{4^i} = \frac{1}{3}$$

Na slici 7.32 dato je intuitivno pojašnjenje prostorne složenosti ove tehnike, a odgovara i originalnom predlogu Vilijamsa za skladištenje MIP mapa. Međutim, pokazuje se da ovo nije najzgodnije jer nema

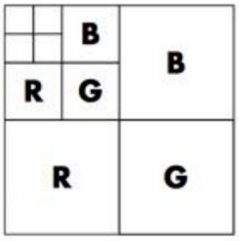


Slika 7.30: Ilustracija uprosečavanja vrednosti tekstura.



Slika 7.31: Piramida MIP mapa.

prostorne lokalnosti – potrebno je čitati tri vrednosti za crvenu, zelenu i plavu komponentu boje iz potpuno različitih regiona slike.



Slika 7.32: Ilustracija skladištenja MIP mapa.

Razmotrimo koliko se brzo menjaju vrednosti u i v koordinate duž x i y ose. U tom cilju računamo dužine vektora L_x i L_y određenih tačkama sa teksture koje odgovaraju susednim pikselima ekrana po x i po y osi (slika 7.33):

$$\frac{du}{dx} = u_{10} - u_{00}, \quad \frac{dv}{dx} = v_{10} - v_{00}$$

$$\frac{du}{dy} = u_{01} - u_{00}, \quad \frac{dv}{dy} = v_{01} - v_{00}$$

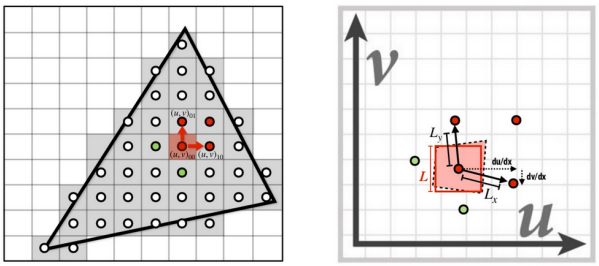
$$L_x^2 = \left(\frac{du}{dx}\right)^2 + \left(\frac{dv}{dx}\right)^2 \quad L_y^2 = \left(\frac{du}{dy}\right)^2 + \left(\frac{dv}{dy}\right)^2$$

Dužine vektora L_x i L_y daju okvirnu procenu koliki region teksture pokriva jedan piksel:

$$L = \sqrt{\max(L_x^2, L_y^2)}.$$

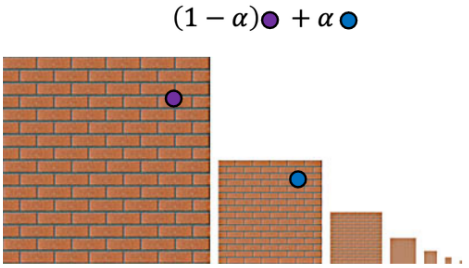
Pošto je kod MIP mapa svaka naredna slika teksture duplo manje dimenzije od prethodne, nivo d MIP mape koji treba koristiti za dati piksel računamo po formuli:

$$d = \log_2 L.$$



Slika 7.33: Procena veličine regiona teksture koji pokriva jedan piksel ekrana.

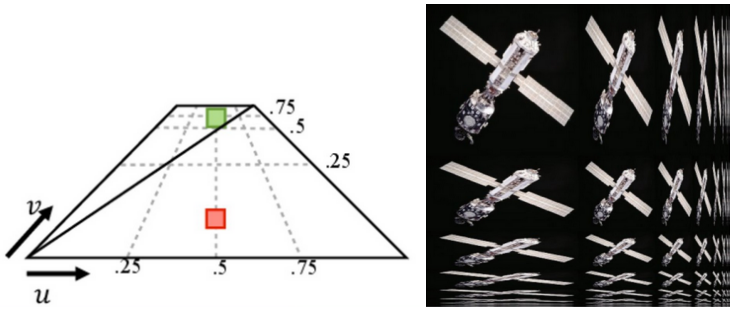
Primetimo da ukoliko bismo za dobijanje vrednosti piksela ekrana koristili odgovarajući teksel nivoa MIP mape koji je najbliži vrednosti d, možemo dobiti efekat pri kom se nivo detaljnosti prikaza naglo menja. Umesto da biramo jedan nivo MIP mape koji odgovara najbližem celom broju, možemo da koristimo neprekidnu vrednost nivoa d. Pošto unapred računamo fiksan broj nivoa MIP mape, možemo vršiti interpolaciju između dva susedna nivoa MIP mape. Na ovaj način dobijamo *trilinearnu interpolaciju*, kojom mešamo vrednosti teksture dva susedna nivoa MIP mape (slika 7.34). Istaknimo da ovo postaje dosta skupo za računanje jer je za svaki piksel ekrana potrebno raditi trilinearnu interpolaciju vrednosti teksture, međutim, to i jeste nešto što grafička kartica ume dobro da radi – da brzo izvršava ovu vrstu operacija.



Slika 7.34: Trilinearna interpolacija.

Primetimo da trilinearno filtriranje pretpostavlja da se uzorci smanjuju istom brzinom duž u i v koordinate. Naime, i bilinearno i trilinearno filtriranje su primeri tzv. *izotropnog filtriranja*. Ovaj mehanizam nije najpogodniji kada brzina promene treba da bude različita duž različitih koordinata. Razmotrimo primer ravni koja je zarotirana u odnosu na kameru – promene duž koordinate v su veće nego duž koordinate u. U ovom scenariju bolji rezultat se postiže korišćenjem *anizotropnog filtriranja* (eng. anisotropic filtering). Kod anizotropnog filtriranja se pravi nova mapa teksture koja nezavisno smanjuje sliku po x i y osi (slika 7.35):

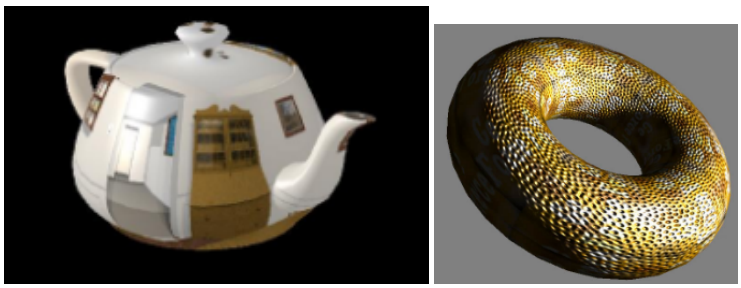
$$(d_x, d_y) = (\log_2 \sqrt{L_x^2}, \log_2 \sqrt{L_y^2})$$



Slika 7.35: Anizotropno filtriranje.

7.3 Preslikavanje normala, neravnina, pomeraja i okruženja

Ideja preslikavanja tekstura kako bi se izmenila boja površi može se i uopštiti. Naime, mnogi atributi procesa renderovanja mogu se zadati u vidu slike: normale površi, transparentnost objekta, boja svetlosti i njen intenzitet, pozicioniranje objekata itd. Tokom godina su, pored osnovne tehnike preslikavanja tekstura, razvijene i složenije vrste preslikavanja kao što je preslikavanje normala kojim se vektori normala menjaju tokom procesa renderovanja i preslikavanje okruženja kod koga se kao mapa teksture koristi slika okruženja čime se omogućava simulacija jako spekularnih površi. Ove vrste preslikavanja su omogućile simulaciju fotorealističnih slika u realnom vremenu redukovanjem broja poligona koji se renderuju i smanjenjem količine izračunavanja potrebne za računanje osvetljenja (slika 7.36).

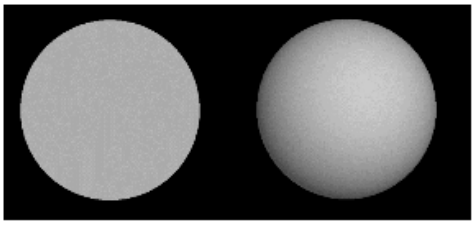


Slika 7.36: Ilustracija preslikavanja okruženja i preslikavanja neravnina (preuzeto sa https://courses.cs.vt.edu/~cs4204/lectures/texture_mapping.pdf.

7.3.1 Preslikavanje normala

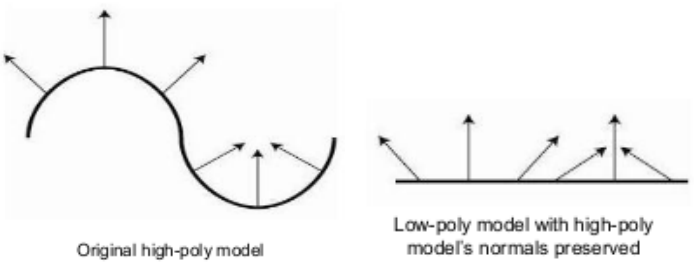
Stvarni objekti imaju sitne detalje i nepravilnosti koje je u praksi teško modelovati. Razmotrimo primer modelovanja pomorandže (slika 7.37). Mogli bismo krenuti od sfere narandžaste boje, ali bismo na taj način dobili previše pojednostavljen i nerealističan model. Ako bismo sferu zamenili modelom koji se sastoji od velikog broja poligona koji potpuno verno opisuju sve detalje modela i koji su odgovarajućih boja i koji imaju potrebna svojstva materijala dobili bismo veoma složen model, a i dalje ne bismo sasvim verno opisali npr. rupice koje postoje na površini pomorandže. Umesto pravljenja složenih geometrijskih modela, mogli bismo fotografiju prave pomorandže nalepiti na model sfere – ovaj postupak odgovara preslikavanju tekstura. Međutim, ni na ovaj način ne bismo dobili sasvim zadovoljavajući prikaz: površ bi i dalje izgledala glatko jer bi njena geometrija ostala neizmenjena. Naime, čovekov vizualni sistem zaključuje kakav je oblik objekta na osnovu toga kako je osvetljen, odnosno na osnovu senčenja objekta (slika 7.38), a osvetljenost tačke na objektu određena je vektorom normale u toj tački.

Slika 7.37: Pomorandža.



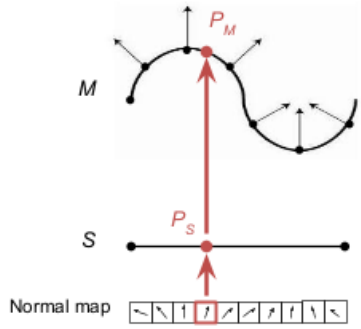
Slika 7.38: Zaključivanje kakvog je oblika objekat na osnovu senčenja (preuzeto iz slajdova Univerziteta Brown).

Dakle, postoji redom potreba da se geometrija modela lokalno izmeni što je moguće izvesti izmenom same geometrije mreže, ali i na drugi način: izmenom vektora normala površi korišćenjem mapa tekstura i ovu tehniku nazivamo *preslikavanje normala* (eng. normal mapping). Preslikavanjem normala emulira se izmena vektora normala tokom procesa renderovanja na osnovu *mape normala* (eng. normal map), čime se menja osvetljenje pojedinačnih tačaka sa površi, dok sama jednačina osvetljenosti ostaje neizmenjena. Naravno, na ovaj način se samo daje privid dubine objekta, jer se sama geometrija modela ne menja.



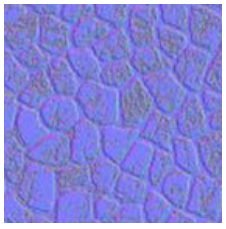
Slika 7.39: Ilustracija postupka kojim se dolazi do mape normala (preuzeto iz slajdova Univerziteta Brown).

Mapu normala možemo dobiti na sledeći način: u fazi pripreme modela razmatramo složeni model koji sadrži veoma veliki broj poligona, izračunamo vektore normala ove mreže i zapamtimo ih u mapi normala. Nakon toga, u fazi obrade možemo znatno redukovati broj poligona u mreži i razmatrati pojednostavljenu mrežu sa malim brojem poligona. Za svaki piksel mape pojednostavljene mreže traži se jedna najbliža odgovarajuća tačka originalne mreže, ili se računa prosek vrednosti susednih normala originalne mreže i odgovarajuće vrednosti se pamte u mapi normala (slika 7.40), a za računanje osvetljenja koristiti vektore normala zapamćene u mapi normala (slika 7.39).



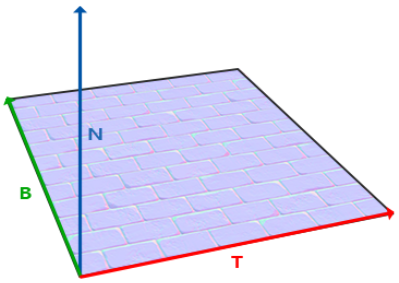
Slika 7.40: Kreiranje mape normala: x, y i z komponenta vektora normale se čuvaju kao R, G i B komponente vrednosti boje teksture (preuzeto iz slajdova Univerziteta Brown).

Čuvanje vektora normala u mapi teksture na prvi pogled ne zvuči intuitivno, jer se u teksturi uobičajeno za svaki uzorak boje čuvaju njena crvena (R), zelena (G) i plava (B) komponenta u vidu vektora dužine 3. Međutim, primetimo da je na isti način umesto tri komponente boje moguće sačuvati x, y i z koordinatu vektora normale. Dakle, mapa normala će za svaki poligon čuvati tačno usmerenje vektora normale. Skoro sve mape normala biće plavičaste boje jer su gotovo svi vektori normala bliski vektoru usmerenom ka pozitivnom delu z koordinatne ose (slika 7.41). Delovi mape normala koji nisu plavičaste boje odgovaraju onim vektorima normale koji odstupaju od ovog trenda i koji daju dubinu teksturama.



Slika 7.41: Vizualizacija mape normala.

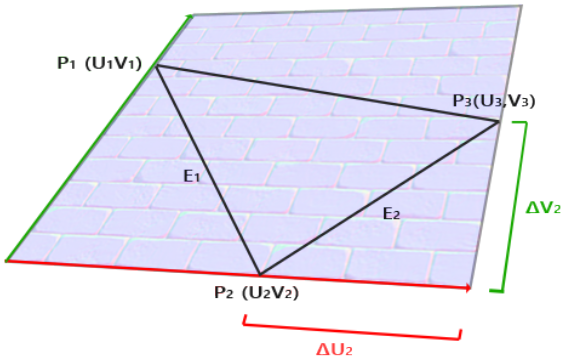
Problem koji se može javiti kada se koristi preslikavanje normala je taj što ukoliko normala ravni na koju primenjujemo preslikavanje normala nije usmerena takođe u pozitivnom delu z ose, osvetljenje neće izgledati dobro. Razlog za to je taj što su vektori normala mape normala i dalje većinom usmereni ka pozitivnom delu z ose, a trebalo bi da prate usmerenje ravni. Ovo se rešava korišćenjem *prostora tangenti* (eng. tangent space) u kome je moguće koristiti uvek istu mapu normala bez obzira na orijentaciju vektora normale ravni, a naknadno se vrši transformacija vektora normala u željeni koordinatni sistem. Prostor tangenti je prostor u kome se koordinate zadaju relativno u odnosu na ravan trougla. S obzirom na to da je mapa normala definisana u prostoru tangenti, potrebno je odrediti matricu kojom se vrši transformacija normala iz prostora tangenti u prostor u kome su vektori normala poravnati sa vektorom normale površi. Ova matrica se naziva *TBN matrica* (akronim od naziva vektora tangente, bitangente i normale). Kako bismo odredili ovu matricu, potrebno je doći do tri međusobno upravna vektora: vektora koji je usmeren naviše, vektora koji je usmeren udesno i vektora koji pokazuje napred. Vektor koji pokazuje naviše nam je poznat – to je vektor normale površi. Druga dva vektora predstavljaju vektore tangente i bitangente (slika 7.42). Oni odgovaraju teksturnim koordinatama površi.



Slika 7.42: Vektori tangente, bitangente i normale za datu površ.

Razmotrimo $\triangle P_1P_2P_3$ sa slike 7.43 čije su teksturne koordinate redom jednake $P_1(u_1, v_1)$, $P_2(u_2, v_2)$ i $P_3(u_3, v_3)$. Vektori stranica $\vec{E_1}$ i $\vec{E_2}$ trougla $P_1P_2P_3$ mogu se izraziti kao:

$$\begin{aligned} \vec{E_1} &= (u_2 - u_1)\vec{T} + (v_2 - v_1)\vec{B} \\ \vec{E_2} &= (u_3 - u_2)\vec{T} + (v_3 - v_2)\vec{B} \end{aligned}$$



Slika 7.43: Izražavanje vektora stranica trougla preko vektora tangente i bitangente.

Primetimo da su svi pomenuti vektori $\vec{E_1}, \vec{E_2}, \vec{T}$ i $\vec{B}$ vektori u trodimenzionom prostoru, odnosno dimenzije su tri. Stoga, prethodne dve jednačine možemo objedinjeno zapisati korišćenjem matrica kao:

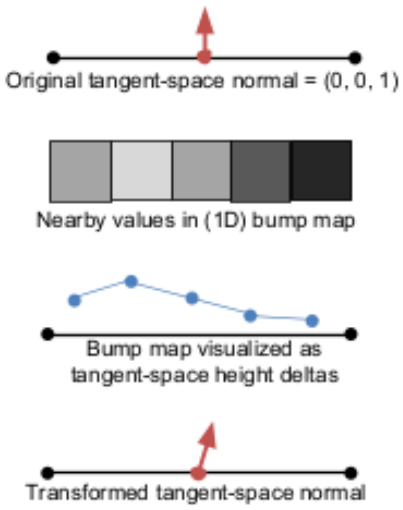
$$\begin{bmatrix} E_{1x} & E_{1y} & E_{1z} \\ E_{2x} & E_{2y} & E_{2z} \end{bmatrix} = \begin{bmatrix} u_2 - u_1 & v_2 - v_1 \\ u_3 - u_2 & v_3 - v_2 \end{bmatrix} \cdot \begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \end{bmatrix}$$

Matrični zapis omogućava da jednostavno izrazimo vektore tangente i bitangente ako su poznate koordinate trougla $P_1P_2P_3$, kao i njegove teksturne koordinate u i v.

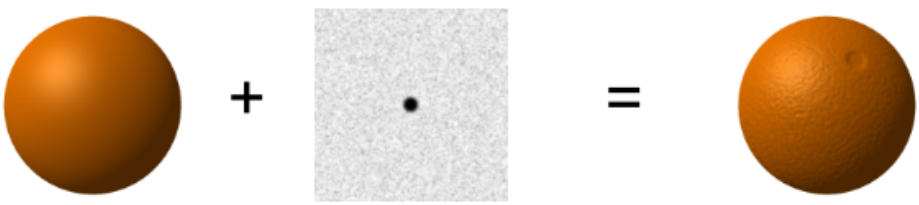
$$\begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \end{bmatrix} = \begin{bmatrix} u_2-u_1 & v_2-v_1 \\ u_3-u_2 & v_3-v_2 \end{bmatrix}^{-1} \cdot \begin{bmatrix} E_{1x} & E_{1y} & E_{1z} \\ E_{2x} & E_{2y} & E_{2z} \end{bmatrix}$$
$$= \frac{1}{(u_2-u_1)(v_3-v_2)-(u_3-u_2)(v_2-v_1)} \begin{bmatrix} v_3-v_2 & -(v_2-v_1) \\ -(u_3-u_2) & u_2-u_1 \end{bmatrix} \cdot \begin{bmatrix} E_{1x} & E_{1y} & E_{1z} \\ E_{2x} & E_{2y} & E_{2z} \end{bmatrix}$$

7.3.2 Preslikavanje neravnina

Umesto kodiranja samih vektora normala, u mapi tekstura se mogu čuvati relativne visine, tj. neravnine tačaka u kojima je potrebno izračunati vektore normala. Na taj način mogu se simulirati uzvišenja i udubljenja na objektu što utiče na jednačinu osvetljenja i daje privid oblika površi na osnovu osvetljenja. Ovu vrstu preslikavanja nazivamo *preslikavanje neravnina* (eng. bump mapping) i nju je 1978. godine po prvi put predložio Džejms Blin. Detalji koji se kreiraju ovom vrstom preslikavanja predstavljaju jednu vrstu obmane jer se preslikavanjem ne povećava rezolucija modela. Dakle, *mapa neravnina* (eng. bump map) sadrži vrednosti na osnovu kojih se modifikuju vrednosti vektora normala (slika 7.44). S ozbirom na to da se visina tačke sa objekta kodira jednim skalarom, vrednosti u mapi neravnina mogu se predstaviti odgovarajućom nijansom sive, tako da crna boja odgovara minimalnoj visini u odnosu na površ, a bela boja maksimalnoj visini. Dakle, crna boja odgovara tome da je površ uvučena, bela da je izdignuta, a srednje siva da je površ teksture na istoj visini kao i sam model (slika 7.45). Mapa neravnina se može dobiti tako što pronađemo teksturu kakvu želimo da simuliramo i onda je konvertujemo u nijanse sive. Za neke slike ovaj pristup daje jako dobar rezultat, dok kod nekih slika dobijeni prikaz nije zadovoljavajući. Mape neravnina predstavljaju jedan od starijih tipova mapa i danas su retko u upotrebi.



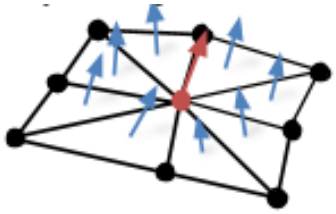
Slika 7.44: Računanje vektora normale na osnovu mape neravnina (preuzeto iz slajdova Univerziteta Brown).



Slika 7.45: Crna tačka na mapi neravnina pravi malo udubljenje na sferi, bez izmene geometrije samog modela (preuzeto iz slajdova Univerziteta Brown).

Vektor normale se na osnovu mape neravnina izračunava tako što se sakupi nekoliko uzoraka visine iz teksture i one se konvertuju u koordinate u trodimenzionom prostoru. Vektor normale u izabranoj

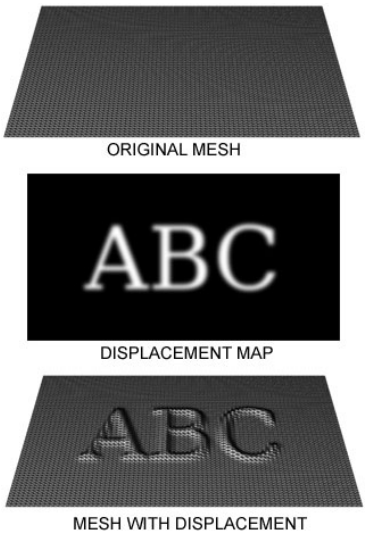
tački predstavlja prosek normala susednih trouglova (slika 7.46). Na slici 7.45 prikazan je primer primene mape neravnina na sferu narandžaste boje.



Slika 7.46: Vektor normale u datoj tački računa se kao prosek vektora normala susednih trouglova (preuzeto iz slajdova Univerziteta Brown).

7.3.3 Prealikavanje pomeraja

Alternativu prethodno pomenutim vrstama preslikavanja predstavlja *preslikavanje pomeraja* (eng. displacement mapping) kojim se vrši pomeranje temena mreže duž vektora normale u toj tački za vrednost koja se čuva u *mapi visina* (eng. height map), odnosno *mapi pomeraja* (eng. displacement map). Razmotrimo primer ravni koja sadrži veliki broj temena i gde se svako teme pomera za onu vrednost koja se nalazi u mapi visina. Dakle, ovim tipom preslikavanja fizički se izmešta mreža na koju se mapa primenjuje. Ako se radi nad temenima u ravni, svako teme se izdiže za vrednost visine iz mape, čime se ravna površ transformiše u površ koja sadrži neravnine (slika 7.47). Ovom vrstom preslikavanja se vrši izmena same geometrije mreže, što će uticati na to da će osvetljenje i senke izgledati jednako tačno kao da je odgovarajući model napravljen ručno. Problem sa ovom vrstom preslikavanja ja taj što je potrebno da ravan sadrži veoma veliki broj trouglova da bi se dobila realistična slika, što računski postaje veoma zahtevno. Naime, preslikavanjem pomeraja se ne dobijaju detalji između temena kao što je to slučaj sa preslikavanjem normala i neravnina. Ovo se može popraviti korišćenjem *preslikavanja paralakse* (eng. parallax mapping) koje ne zahteva korišćenje dodatnih temena, već koristi neke napredne tehnike da zavara oko posmatrača, koje ovde nećemo analizirati.

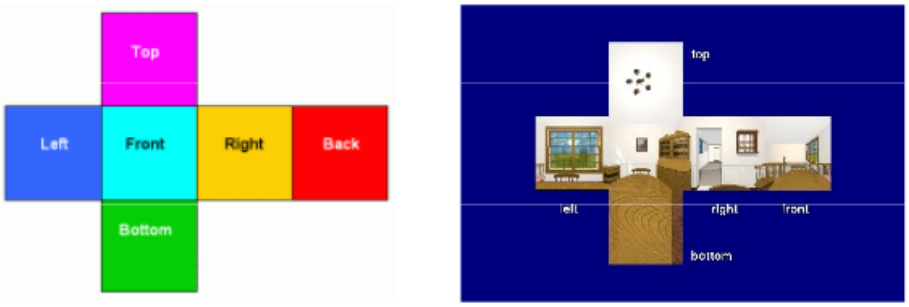


Slika 7.47: Primena mape pomeraja na mrežu (preuzeto sa vikipedije).

7.3.4 Preslikavanje okruženja

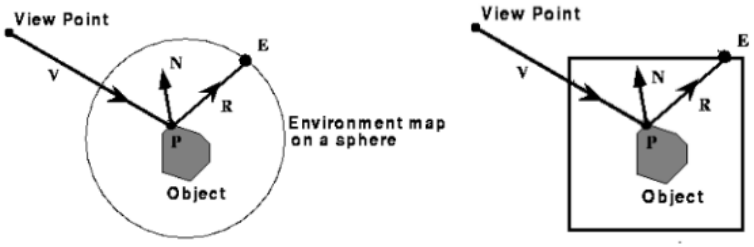
Precizno simuliranje refleksije okruženja na sjajnim objektima zahteva primenu rekurzivnog rejtrejsing algoritma što je prilično skupo. Umesto toga, može se koristiti *preslikavanje okruženja* (eng. environment mapping), odnosno *preslikavanje refleksija* (eng. reflection mapping) koje predstavlja jednostavan a dovoljno moćan mehanizam za generisanje refleksija. Preslikavanje okruženja se može primeniti na površi koje se nalaze u zatvorenom prostoru i ono omogućava simulaciju visoko reflektujućih površi.

Ovaj mehanizam su prvi predložili Džim Blin i Martin Njuel 1976. godine. Pretpostavimo da je okruženje veoma daleko od objekta. Unutar scene koja okružuje sjajni objekat puštamo zrak refleksije u sliku scene koja ga okružuje, tj. *mapu refleksije* (eng. reflection map), nalazimo odgovarajuću tačku sa mape i koristimo njenu vrednost da bismo obojili tačku sa površi objekta. U originalnom algoritmu je za okruživanje scene korišćena sfera, međutim pošto se pri radu sa sferom javlja određen broj problema kao što su distorzija slike i zavisnost od pozicije posmatrača, Nejt Grin (Nate Green) je 1986. godine predložio da se umesto sfere koristi model kocke (slika 7.48).



Slika 7.48: Korišćenje kocke za mapu okruženja (preuzeto sa http://web.cse.ohio-state.edu/~wang.3602/courses/cse5542-2013-spring/17-env.pdf).

Da bismo odredili odgovarajuću vrednost mape okruženja koju treba prikazati u tački P sa površine reflektujućeg objekta, računamo vektor $\overrightarrow{V}$ iz tačke P do oka posmatrača, a zatim i zrak refleksije $\overrightarrow{R}$ kao vektor simetričan vektoru $\overrightarrow{V}$ u odnosu na vektor $\overrightarrow{N}$ normale površi u tački P (slika 7.49). Određujemo presečnu tačku vektora $\overrightarrow{R}$ sa mapom okruženja i nju koristimo za tačku P. Tačka P se tretira kao centar mape, odnosno mapa okruženja se pomera tako da tačka P bude u centru (efektivno, pretpostavljamo da je okruženje u kome se objekat nalazi mnogo veće nego što je zaista, odnosno da je mapa beskonačno daleko od objekta).



Slika 7.49: Računanje vrednosti u datoj tački P na osnovu mape okruženja ako se koristi sfera (levo) ili kocka (desno).

Mapa okruženja se može kreirati renderovanjem kompletne scene iz centra objekta, po jednom za svaku od šest strana kocke. U slučaju kada je scena statična ovo se može uraditi samo jednom, međutim, u slučaju objekata koji se kreću potrebno je mapu okruženja kreirati dinamički, u vreme izvršavanja, što može biti vremenski veoma zahtevno. Da bi se ovo donekle ublažilo u praksi se koriste mnoge dodatne tehnike.

7.4 Pitanja

7.1 Šta su teksture?

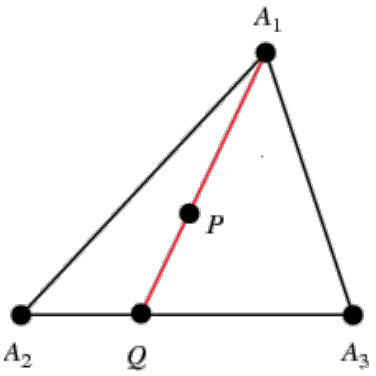
7.2 Iz kog intervala uzimaju vrednosti teksturne koordinate u i v?

7.3 Kada je pogodnije koristiti skaliranje, a kada popločavanje prilikom nanošenja tekstura?

7.4 Šta treba da važi za sliku teksture kod popločavanja?

7.5 Za šta koristimo baricentrične koordinate?

7.6 Koje su baricentrične koordinate tačke P prikazane na slici u odnosu na temena A_1, A_2 i A_3 trougla, ako je $A_2Q : QA_3 = 1 : 2$ i $A_1P : PQ = 3 : 2$?



7.7 Koja dva svojstva važe za baricentrične koordinate neke tačke u odnosu na dati trougao?

7.8 Kako možemo dokazati da su vrednosti sve tri baricentrične koordinate između 0 i 1?

7.9 Na koji način se vrši nanošenje teksture na složeno geometrijsko telo?

7.10 Od koja dva koraka se sastoji dvostepeno preslikavanje tekstura?

7.11 Na koja tri načina je moguće realizovati preslikavanje teksture sa graničnog opsega objekta na sam objekat?

7.12 Šta sadrži mapa normala? Na koji način se može dobiti?

7.13 Čemu služi preslikavanje neravnina?

7.14 Uporediti preslikavanja normala i preslikavanje neravnina.

7.15 Kako funkcioniše preslikavanje pomeraja?

7.16 Koji efekat se postiže preslikavanjem okruženja?

7.17 Koji se geometrijski objekti najčešće koriste kao mape okruženja?

Glava 8

Osvetljenje, senčenje i senke

Scena koju iscrtavamo u interaktivnoj računarskoj grafici sastoji se od objekata, kamere i izvora svetla. Na sceni može postojati jedan ili više izvora svetla. Za svaki izvor svetla potrebno je zadati njegove karakteristike: položaj na sceni, oblik, pravac u kome emituje svetlost, boju svetlosti koju emituje i dr. Odbijanje svetlosti od objekata na sceni određeno je karakteristikama svetlostima, svojstvima materijala i geometrijom tog objekta. Potrebno je modelovati ponašanje po kome svaka tačka sa površine objekta prima svetlost direktno od izvora svetla koji nisu blokirani drugim objektima i indirektno od svetlosti koja dolazi od drugih objekata na sceni nakon refleksije i prelamanja svetlosti. Složeni algoritmi zasnovani na pravilima fizike koji modeluju rekurzivnu prirodu međuobjektnih refleksija zahtevaju mnogo izračunavanja. Kada je cilj prikaz u realnom vremenu, ovi algoritmi često zahtevaju više procesorske moći nego što današnji hardver može da pruži. Stoga se računarska grafika u realnom vremenu uglavnom zasniva na tehnikama aproksimacije. Sa današnjim grafičkim procesorima moguće je napraviti dovoljno dobre aproksimacije kojima je moguće zavarati čovekov vizulalni sistem. U daljem tekstu razmotrićemo osnovne modele osvetljenja i senčenja koji nisu fizički zasnovani ali daju prihvatljiv vizuelni izgled u razumnom vremenu i dovoljno su efikasni da se mogu koristiti u aplikacijama koje rade u realnom vremenu. U novije vreme, sa brzim razvojem grafičkih procesorskih jedinica, aproksimacije višeg kvaliteta u realnom vremenu su postale dosežnije, kao i mogućnost da simuliramo fiziku interakcija svetlosti.

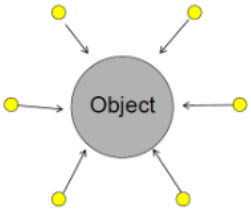
Problem osvetljenja (eng. illumination problem) odnosi se na izračunavanje intenziteta i boje pojedinačne tačke sa površi na sceni onako kako je posmatrač doživljava, simuliranjem atributa svetlosti. Osvetljenje je funkcija geometrije scene (koja uključuje modele na sceni, svetla i kameru i njihove prostorne odnose) i parametara materijala objekata (refleksije, apsorpcije itd.). *Problem senčenja* (eng. shading problem) primenjuje model osvetljenja na skup tačaka i boji kompletnu površ. Dakle, model senčenja predstavlja širi okvir i on koristi model osvetljenja. Naglasimo da senčenje, iako naziv može asocirati na to, nije ni u kakvoj vezi sa određivanjem senki. Neki modeli senčenja pozivaju model osvetljenja za svaki pojedinačni piksel slike, dok drugi pozivaju model osvetljenja samo za neke piksele, dok se za preostale koristi interpolacija. Primetimo da su problemi osvetljenja i senčenja znatno komplikovaniji od problema vidljivosti. Naime, prilikom razmatranja vidljivosti jedini uslov koji je potrebno ispitati jeste da li postoji prepreka između dve tačke čiju vidljivost ispitujemo. Interakcija između svetala i površi je pak nešto složenija.

Postoji više različitih modela osvetljenja i modela senčenja. Neka od rešenja problema osvetljenja zasnovana su na iskustvu i eksperimentima i nisu utemeljena u fizici, ali daju dobre rezultate.

8.1 Tipovi izvora svetla na sceni

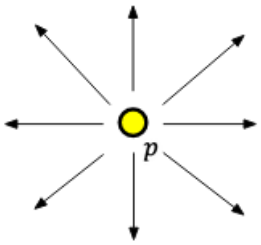
Postoji više različitih tipova izvora svetla koja se mogu naći na sceni. U realnom svetu svaki izvor svetla je trodimenzioni objekat koji ima neku geometriju. U kontekstu računarske grafike razmatraćemo pojednostavljene, idealizovane modele izvora svetla i to: ambijentalno svetlo, tačkaste izvore svetla, direkcione izvore svetla, proširene izvore svetla i spot svetla.

Ambijentalno svetlo (eng. ambient light), kao što i samo ime sugeriše, je svetlo bez usmerenog izvora koje podjednako utiče na sve objekte na sceni i proizvod je višestrukog odbijanja svetlosti od svih površina prisutnih u okruženju. Svetlost koja potiče od ambijentalnog svetla se jednako širi u svim smerovima i po svim objektima, nezavisno od toga gde se objekat nalazi na sceni (slika 8.1). Jedini parametar ambijentalnog svetla je njegov intenzitet I.



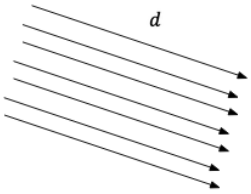
Slika 8.1: Ilustracija ambijentalnog svetla.

Tačkasti izvor svetla (eng. point light source) iz jedne tačke ravnomerno širi zrake u svim smerovima. Za razliku od ambijentalnog svetla, pored intenziteta svetla I, važan parametar tačkastog izvora svetla je njegova pozicija p na sceni (slika 8.2). Tačkasti model svetla predstavlja dobru aproksimaciju u situaciji kada je izvor svetla značajno manji od veličine objekata na sceni, na primer ako se u prostoriji nalazi mala sijalica.



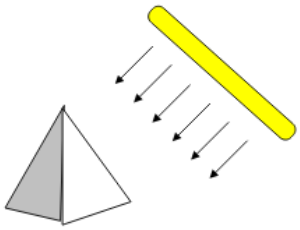
Slika 8.2: Ilustracija tačkastog izvora svetla.

Direkcioni izvor svetla (eng. directional light source) simulira svetlost koja se emituje iz izvora svetla koji je beskonačno daleko od objekata sa scene. Primer direkcionog izvora svetla bi mogla biti sunčeva svetlost. Direkcioni izvor svetla nema konkretnu poziciju, već se za njega zadaje smer u kome zraci svetlosti padaju na objekte sa scene (slika 8.3).



Slika 8.3: Ilustracija direkcionog izvora svetla.

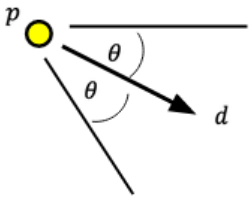
Za razliku od tačkastih izvora svetla, *prošireni* ili *distribuirani izvori svetla* (eng. extended, distributed light sources) modeluju izvor svetla kao trodimenzioni objekat koji ima svoju površinu (slika 8.4). Kao posledica toga, ovi izvori svetla daju mekše senke koje sadrže oblasti samo delimično blokirane od izvora svetla.



Slika 8.4: Ilustracija proširenog izvora svetla.

Za razliku od tačkastog izvora svetla koji se aproksimira tačkom u trodimenzionom prostoru iz koje se svetlost emituje jednako u svim pravcima, *spot svetlo* (eng. spotlight) emituje svetlost samo unutar ograničenog potprostora u obliku kupe. Jedan model spot svetla predstavlja reflektor na sceni (slika 8.5).

Slika 8.5: Reflektor kao model spot svetla.



Slika 8.6: Ilustracija spot svetla.

Spot svetlo se zadaje pozicijom p na sceni, intenzitetom svetla I, smerom d u kome emituje svetlost (eng. spot direction) i uglom odsecanja Θ (eng. cutoff angle) (slika 8.6). Svetlost se emituje iz izvora svetlosti u smerovima koji sa smerom emitovanja svetlosti obrazuju ugao manji od ugla odsecanja. Intenzitet svetlosti uobičajeno opada unutar ovog prostora sa povećanjem udaljenosti zraka od smera emitovanja svetlosti. Ova zavisnost može se opisati kosinusom ugla α između zraka koji razmatramo i smera emitovanja svetlosti. Dakle, u slučaju kada je ugao α manji od ugla odsecanja, intenzitet spot svetla može se opisati jednačinom

$$I = I_{spot} \cdot (\cos \alpha)^e$$

gde je I_{spot} intenzitet spot svetla, a e eksponent kojim se kontroliše brzina kojom intenzitet svetla opada.

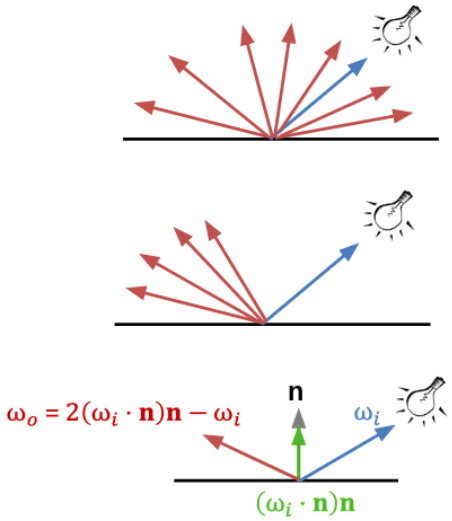
U nastavku teksta ćemo se baviti ambijentalnim, tačkastim i direkcionim izvorima svetla.

8.2 Modelovanje materijala objekata

Prilikom zadavanja svojstava objekata na sceni, potrebno je posebnu pažnju obratiti na modelovanje materijala od koga su sačinjeni objekti. Naime, kada zrak svetlosti padne na neku tačku sa površi način na koji se vrši odbijanje zraka razlikovaće se ako je ta površ visoko reflektujuća, poput površine ogledala, odnosno ako je u pitanju neki difuzni materijal, poput zida. Razlikujemo tri osnovna tipa materijala: difuzne, spekularne i savršeno reflektujuće materijale (slika 8.7). Kod difuznih, odnosno lambertovskih materijala svetlost se nakon što pogodi tačku sa objekta rasipa jednako po svim smerovima. Kod spekularnih (sjajnih) materijala svetlost se rasipa tešnje oko jednog smera – smera koji je simetričan upadnom zraku svetlosti $\overrightarrow{\omega_o}$ u odnosu na vektor normale $\overrightarrow{n}$. Kod savršeno reflektujućih materijala poput ogledala sva svetlost se odbija u tačno jednom smeru – smeru $\overrightarrow{\omega_0} = 2(\overrightarrow{\omega_i} \cdot \overrightarrow{n})\overrightarrow{n} - \overrightarrow{\omega_i}$.

8.3 Osnovni modeli osvetljenja

Postoji više različitih modela osvetljenja, a mi ćemo u nastavku teksta detaljno razmotriti Fongov i Blin-Fongov model osvetljenja koji nisu fizički zasnovani, ali daju vizualno prihvatljive rezultate. Pored njih postoje i mnogi drugi složeniji modeli osvetljenja koji jesu fizički zasnovani kao što su Kuk-Torensov, Torens-Sperouov, Blinov, Oren-Najarov i Vordov model.



Slika 8.7: Interakcija svetlosti i objekta za tri osnovna tipa materijala u računarskoj grafici.

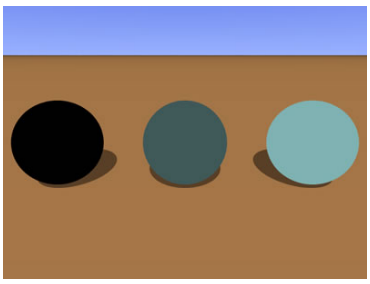
Kako bismo račun učinili jednostavnijim razmatraćemo, pre svega, monohromatsku svetlost, tj. u jednačinama koje budemo razmatrali biće relevantan samo intenzitet svetla.

8.3.1 Ambijentalna refleksija

Ambijentalno svetlo se u računarskoj grafici koristi za simulaciju prirodnog svetla na sceni i dodavanje realističnosti sceni. Ono ne dolazi od nekog konkretnog izvora svetla, već osvetljava sve objekte na sceni podjednako. Ako pretpostavimo da se svetlost rasprostire jednako u svim smerovima i po svim objektima nezavisno od lokacije površi i njene orijentacije, jednačina osvetljenja objekta jednaka je:

$$I = I_a k_a$$

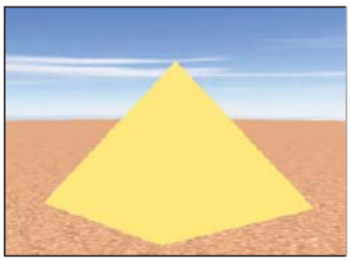
gde je I_a intenzitet ambijentalnog svetla (konstantan za sve objekte), a k_a je *koeficijent ambijentalne refleksije objekta* (eng. ambient-reflection coefficient). Koeficijentom ambijentalne refleksije zadaje se količina ambijentalne svetlosti koja se reflektuje od površi objekta i on uzima vrednost između 0 i 1. Vrednost 0 odgovara veoma tamnim, a 1 veoma svetlim površima (slika 8.8). Koeficijent ambijentalne refleksije se odnosi na materijal od kog je objekat i predstavlja empirijsku pogodnost koja ne odgovara nekom fizičkom svojstvu pravog materijala.



Slika 8.8: Razlika u vrednostima koeficijenta ambijentalne refleksije. Sleva nadesno vrednosti su redom jednake 0.0, 0.5 i 1.0.

Na slici 8.9 dat je prikaz piramide koja je osvetljena samo ambijentalnim svetlom. Možemo primetiti da su sve tačke piramide osvetljene istim intenzitetom svetlosti i da se ne uočava prelaz sa jedne strane piramide na drugu.

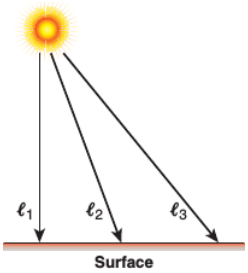
Ambijentalno svetlo obezbeđuje da je svaka površ na sceni osvetljena do nekog stepena, čime se sprečava dobijanje nerealističnih tamnih regiona na slici u slučaju površi koje su daleko od izvora svetla. Kada je u kombinaciji sa drugim tipovima svetla, količina ambijentalnog svetla se postavlja na minimum.



Slika 8.9: Piramida osvetljena samo ambijentalnim svetlom.

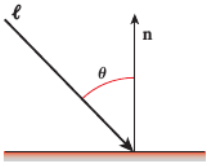
8.3.2 Difuzna refleksija

Razmotrimo situaciju kada se na sceni nalazi tačkasti izvor svetla. Za razliku od ambijentalnog svetla, kod koga je osvetljenost kompletnog objekta konstantna, kod tačkastog izvora svetla osvetljenost objekta varira od jednog dela objekta do drugog. Naime, osvetljenost tačke na objektu zavisi od ugla pod kojim svetlost pada na objekat (slika 8.10).



Slika 8.10: Zraci koji potiču od tačkastog izvora svetla padaju na ravnu površ pod različitim uglovima.

Difuzna (lambertovska) refleksija (eng. diffuse reflection) je refleksija svetlosti od matiranih, hrapavih površina. Ovakve površine izgledaju jednako osvetljene iz svih uglova posmatranja. Osvetljenost takvih površi ne zavisi od pozicije posmatrača, već isključivo od toga koliko direktno svetlost pada na površ: to se meri uglom θ između pravca svetla $\overrightarrow{L}$ i vektora normale površi $\overrightarrow{N}$. Naime, što je ovaj ugao manji to je osvetljenost veća. Ovu zavisnost možemo modelovati kosinusom ugla θ koji ova dva vektora zahvataju (slika 8.11).



Slika 8.11: Osvetljenost matiranih površi meri se uglom θ koji zahvataju pravac svetla i vektor normale površi.

Osvetljenost u ovom modelu se opisuje jednačinom:

$$I = I_p k_d \cos\theta$$

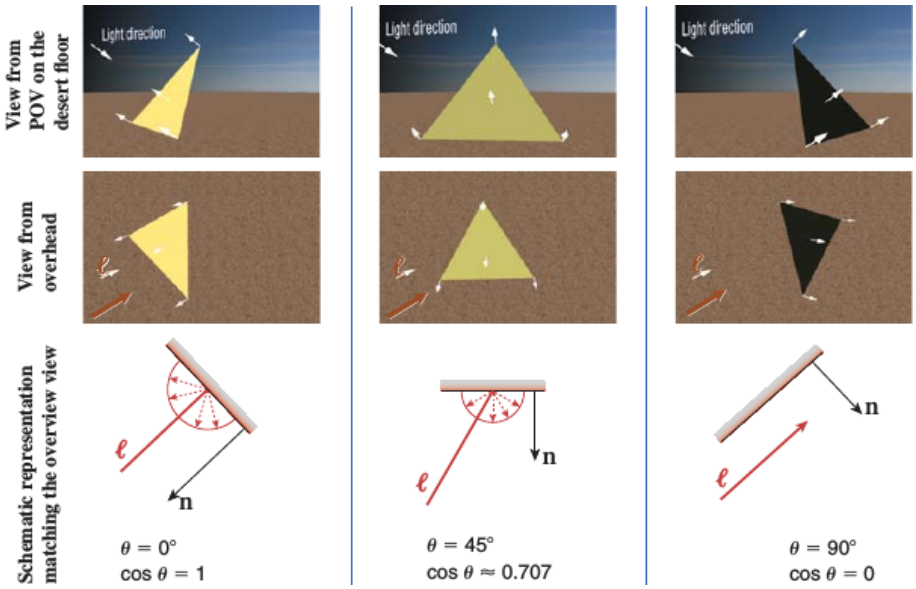
gde je I_p intenzitet tačkastog izvora svetla, k_d *koeficijent difuzne refleksije* (eng. diffuse-reflection coefficient) materijala od kojeg je načinjen objekat, a θ ugao između pravca svetla $\overrightarrow{L}$ i vektora normale površi $\overrightarrow{N}$. Koeficijent difuzne refleksije k_d ima vrednost između 0 i 1 i za visoko reflektivne površine uzima vrednost blisku 1, dok je za površi koje apsorbuju najveći deo svetlosti vrednost k_d bliska vrednosti 0. Ova jednačina se često naziva i *Lambertovo kosinusno pravilo* (eng. Lambert's cosine law) (slika 8.12).

Ako su vektori $\overrightarrow{L}$ i $\overrightarrow{N}$ jediničnog intenziteta, onda je:

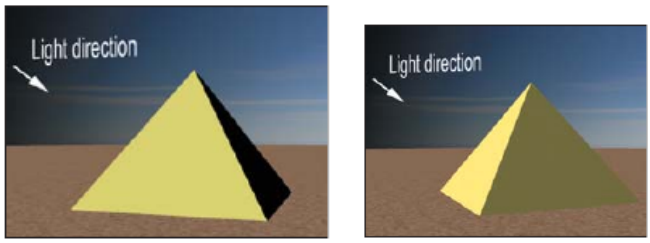
$$I = I_p k_d \cos\theta = I_p k_d (\overrightarrow{L} \cdot \overrightarrow{N})$$

Razmotrimo direkcioni izvor svetla. Ukoliko bi se osvetljenost objekata na sceni računala po formuli

$$I = I_p k_d (\overrightarrow{L} \cdot \overrightarrow{N})$$



Slika 8.12: Osvetljenost računata na osnovu Lambertovog kosinusnog pravila za tri različite vrednosti ugla θ.



Slika 8.13: Renderovanje piramide sa direkcionim izvorom svetla, pri čemu je ugao između pravca svetla i vektora normale najdesnije strane koja je vidljiva blizak a) 90^0 b) 70^0.

objekti bi izgledati kao da su osvetljeni samo sa jedne strane i da se nalaze u mračnoj prostoriji (slika 8.13). Kako bi se taj efekat ublažio, često se koristi sledeća jednačina koja uključuje i ambijentalnu komponentu osvetljenosti:

$$I = I_a k_a + I_p k_d (\overrightarrow{L} \cdot \overrightarrow{N})$$

8.3.3 Slabljenje izvora svetla

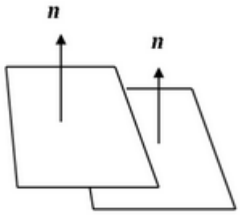
Pretpostavimo da je osvetljenje modelovano kao kombinacija ambijentalne i difuzne komponente. Ukoliko se projekcije dve paralelne površi od istog materijala na slici preklapaju, neće moći da se odredi gde prestaje jedna, a počinje druga, ma koliko se razlikovala njihova rastojanja od izvora svetla (slika 8.14). Kako bi se ovaj problem rešio uzima se u razmatranje i tzv. *faktor slabljenja izvora svetla* (eng. light-source attenuation factor) f_{att} uz koji jednačina osvetljenosti postaje:

$$I = I_a k_a + f_{att} I_p k_d (\overrightarrow{L} \cdot \overrightarrow{N})$$

Najjednostavnija forma faktora slabljenja izvora svetla data je formulom:

$$f_{att} = \frac{1}{d_L^2}$$

gde je d_L rastojanje objekta do izvora svetla. Ova veza proizilazi iz zakonitosti po kojoj energija iz tačkastog izvora svetlosti opada kao inverz kvadrata rastojanja od izvora svetla. Na ovaj način objekti koji su udaljeniji od izvora svetla prikazuju se kao tamniji, odnosno paralelne površi koje se preklapaju, a na različitim rastojanjima su od svetlosti biće međusobno razlučive na slici. Međutim, ova formula ne daje željeni vizuelni efekat jer kada je izvor svetla daleko od objekata na sceni vrednosti $\frac{1}{d_L^2}$ se ne razlikuju previše, dok kada je izvor svetla blizu vrednosti mnogo više variraju.



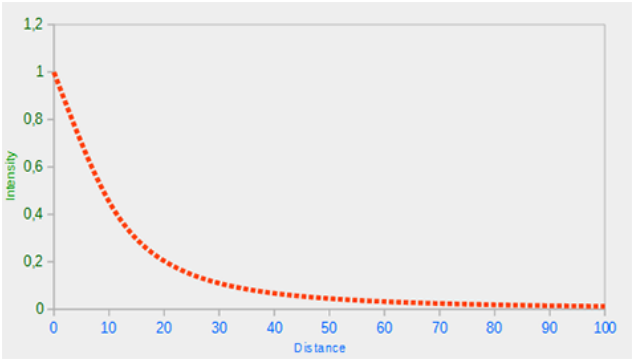
Slika 8.14: Dve paralelne površi od istog materijala koje su različito udaljene od izvora svetla.

Finiji model faktora slabljenja izvora svetla opisan je jednačinom:

$$f_{att} = \min\{\frac{1}{c_1 + c_2 d_L + c_3 d_L^2}, 1\}$$

gde su c_1, c_2 i c_3 korisnički definisane konstante pridružene izvoru svetla (slika 8.15). Vrednosti c_i se podešavaju tako da se dobije željeni vizuelni prikaz. Na primer, kada je vrednost d_L mala, konstanti c_1 se može pridružiti veća vrednost, kako vrednost f_{att} ne bi bila isuviše velika. Gornje ograničenje vrednosti f_{att} je postavljeno na 1 da bi vrednost f_{att} bila između 0 i 1 tako da predstavlja procenat originalnog svetla koji se koristi za bojenje piksela.

Ovaj model se često koristi kada je izvor svetla u tački pogleda jer se na taj način modeluje i slabljenje izvora svetlosti sa povećanjem rastojanja objekta od izvora svetlosti, ali i slabljenje sa rastojanjem od posmatrača.



Slika 8.15: Grafik opadanja intenziteta svetlosti sa povećanjem udaljenosti od izvora svetla (preuzeto sa https://learnopengl.com/Lighting/Light-casters).

Primetimo da u slučaju direkcionog izvora svetla važi $f_{att} = 1$ jer je on na beskonačnom rastojanju od objekata.

8.3.4 Spekularna refleksija

Spekularna refleksija se može uočiti na svakom glatkom, sjajnom objektu. Naime, ako osvetlimo neki objekat jakom belom svetlošću, deo objekta biće veoma svetao i objekat će u tom delu imati belu boju, dok će ostatak objekta biti obasjan difuznim svetlom (slika 8.16).

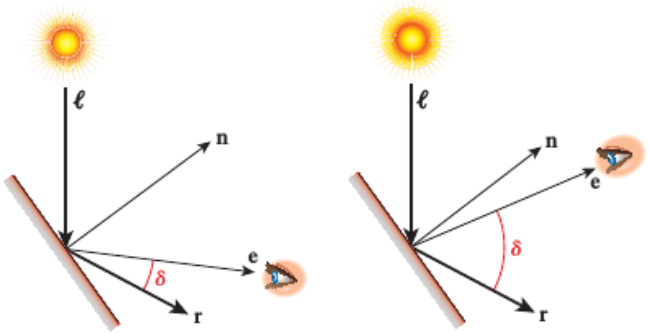
Ako se pomerimo u odnosu na objekat, primetićemo da se sjajni deo objekta takođe izmestio. To je zato što sjajni objekti reflektuju svetlost različito u različitim pravcima. Naime, efekat spekularne refleksije je najjači u pravcu $\overrightarrow{R}$ koji je simetričan pravcu svetlosti $\overrightarrow{L}$ u odnosu na normalu površi $\overrightarrow{N}$, a opada sa povećanjem ugla u odnosu na vektor $\overrightarrow{R}$ (slika 8.17). Štaviše, na savršeno glatkom objektu poput ogledala, ovaj efekat javlja se samo u smeru vektora $\overrightarrow{R}$. Dakle, efekat spekularne refleksije zavisi od pravca posmatranja $\overrightarrow{V}$. Doprinos spekularne refleksije ukupnoj osvetljenosti je prema Fongovom modelu osvetljenja (koji će biti naknadno uveden) jednak:

$$I_{spec} = I_p k_s \cos^s \delta$$

gde je sa δ označen ugao između vektora $\overrightarrow{R}$ i $\overrightarrow{V}$, s je *eksponent spekularne refleksije* (eng. specular-reflection exponent) materijala od koga je objekat napravljen, a k_s *koeficijent spekularne refleksije* (eng.

Slika 8.16: Ilustracija spekularne refleksije – direktne refleksije izvora svetla na sjajnoj površi.

specular-reflection exponent) materijala od koga je objekat. Na ovaj način modeluje se efekat po kome spekularna refleksija ima maksimalnu vrednost kada je ugao δ jednak 0, a zatim naglo opada sa povećanjem vrednosti ugla. Koeficijent spekularne refleksije uzima vrednost između 0 i 1 i njegova vrednost se utvrđuje eksperimentalno tako da proizvede vizuelno prihvatljiv prikaz. Što je ugao δ manji, to je efekat spekularne refleksije veći. Eksponent spekularne refleksije s može imati vrednost od 1 do nekoliko stotina za veoma sjajne površi (slika 8.18). Kod veoma sjajnih površi je vrlo oštro smanjenje spekularnog efekta, dok je kod površi koje nisu toliko sjajne ovaj efekat nešto slabiji, ali je vidljiv na široj površini objekta (slika 8.19). Ovaj model odgovara objektima koji nisu savršeno glatki i ne reflektuju svetlost savršeno, kao što je recimo jabuka ili plastična kašičica.



Slika 8.17: Računanje spekularne refleksije u situaciji kada je pravac posmatranja a) blizu zraka refleksije b) daleko od zraka refleksije; značajna razlika u vrednosti izraza $\cos\delta$ postaje još veća kada se digne na veliki stepen te je stoga u drugom slučaju spekularni doprinos blizak nuli.

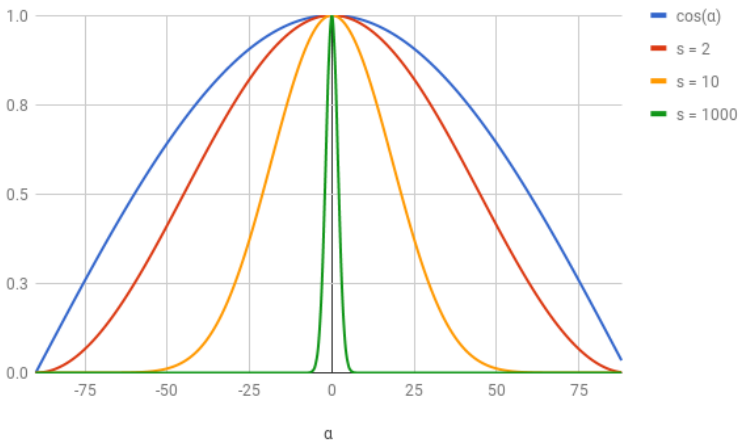
Vektor $\overrightarrow{R}$ se može jednostavno odrediti na osnovu vektora $\overrightarrow{L}$ i vektora $\overrightarrow{N}$. Naime, s obzirom na to da su vektori $\overrightarrow{L}$ i $\overrightarrow{N}$ jedinične dužine projekcija vektora $\overrightarrow{L}$ na pravac vektora $\overrightarrow{N}$ jednaka je $\overrightarrow{L} \cdot \overrightarrow{N}$. Vektor $\overrightarrow{R}$ zahvata isti ugao sa vektorom normale $\overrightarrow{N}$ kao vektor $\overrightarrow{L}$, te je i projekcija vektora $\overrightarrow{R}$ na pravac vektora $\overrightarrow{N}$ jednaka $\overrightarrow{L} \cdot \overrightarrow{N}$. Važi:

$$\overrightarrow{R} + \overrightarrow{L} = 2(\overrightarrow{L} \cdot \overrightarrow{N})\overrightarrow{N}$$

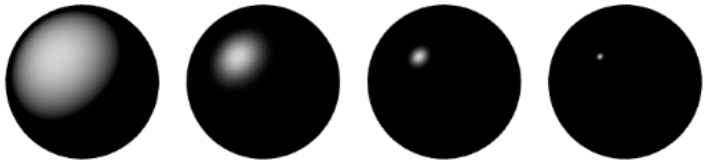
odnosno važi:

$$\overrightarrow{R} = 2(\overrightarrow{L} \cdot \overrightarrow{N})\overrightarrow{N} - \overrightarrow{L}$$

Količina svetla koja će biti reflektovana i način na koji će biti reflektovana u mnogome zavisi od glatkosti površi. Kada su nepravilnosti same površi manje od talasne dužine upadne svetlosti, sva svetlost se jednako reflektuje. Međutim, u realnom svetu to često nije slučaj i kod objekata se javlja difuzna refleksija kojom se upadna svetlost reflektuje u različitim smerovima (slika 8.20). Kod glatkih površi kao što je ogledalo ili mirna površina vode javlja se spekularna refleksija, dok se difuzna refleksija javlja kod hrapavijih površina poput komada odeće ili zida. Naime, ako se posmatrač pomeri u odnosu na zid, njegova boja neće se promeniti: ovo je zato što je svetlost reflektovana od zida ista, bez obzira na to odakle ga posmatramo. U oba pomenuta tipa refleksije svaki pojedinačni zrak prati pravilo refleksije.

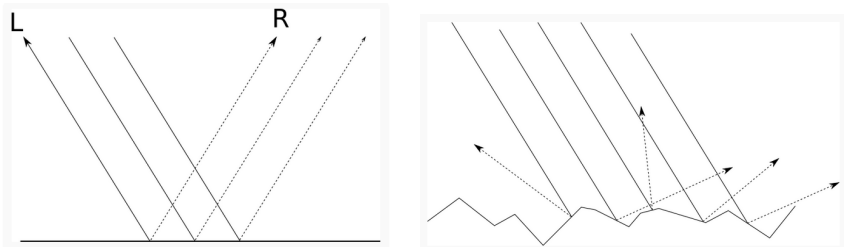


Slika 8.18: Fongova kriva osvetljenosti za različite vrednosti eksponenta spekularne refleksije s: vrednost s raste sleva nadesno (preuzeto sa https://www.gabrielgambetta.com/computer-graphics-from-scratch/light.html).



Slika 8.19: Prikaz spekularne refleksije za različite vrednosti eksponenta spekularne refleksije s.

Ipak, hrapavost materijala podrazumeva da svaki pojedinačni zrak pogađa površ u tački u kojoj se vektor normala razlikuje: efekat ovoga je da se reflektujući zraci od hrapave površi razbacuju u različitim pravcima.



Slika 8.20: Razlika između spekularne i difuzne refleksije.

8.3.5 Fongov model osvetljenja

1975. godine Fong (Bui Tuong Phong) je predložio model osvetljenja, koji je po njemu dobio ime *Fongov model osvetljenja*. Prema ovom modelu ukupna osvetljenost se izražava kao kombinacija ambijentalne, lambertovske i spekularne komponente, sumiranjem za sve izvore svetla:

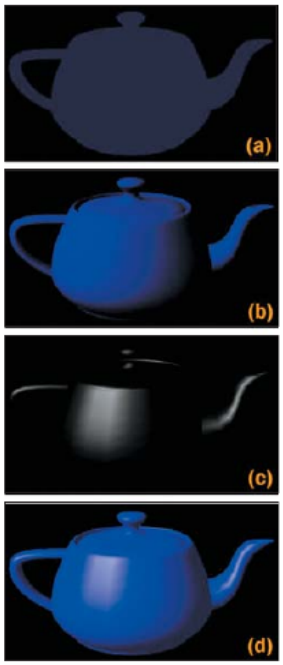
$$I = I_{amb} + I_{diff} + I_{spec}$$

$$I = I_a k_a + f_{att} I_p (k_d \cos\theta + k_s \cos^s \delta)$$

Ako su vektori $\overrightarrow{L}$, $\overrightarrow{N}$, $\overrightarrow{R}$ i $\overrightarrow{V}$ normalizovani, onda važi:

$$I = I_a k_a + f_{att} I_p (k_d (\overrightarrow{L} \cdot \overrightarrow{N}) + k_s (\overrightarrow{R} \cdot \overrightarrow{V})^s)$$

Fongov model osvetljenja je empirijski model i nije fizički zasnovan, ali daje realistični prikaz (slika 8.21). On ne uzima u obzir međuobjektne refleksije te u užem smislu ne predstavlja model globalnog os-



Slika 8.21: Renderovanje čajnika uz prikaz doprinosa svake od tri komponenti koje učestvuju u Fongovoj jednačini osvetljenosti: a) ambijentalna b) difuzna c) spekularna d) rezultat dobijen sumiranjem svih doprinosa.

vetljenja. Međutim, on grubo simulira međuobjektne refleksije dodavanjem ambijentalne komponente jednačini osvetljenja.

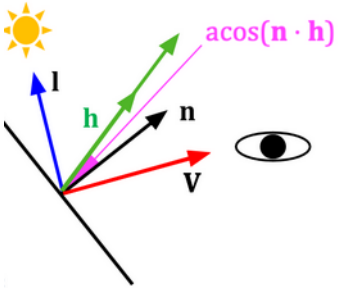
8.3.6 Blin-Fongov model osvetljenja

Pored Fongovog modela osvetljenja, u upotrebi je i *Blin-Fongov model osvetljenja*. On predstavlja modifikaciju Fongovog modela predloženu od strane Džima Blina (Jim Blinn). Razlika između ova dva modela osvetljenja jeste u načinu računanja spekularne komponente. Naime, umesto ugla između pravca reflektovane svetlosti $\overrightarrow{R}$ i pravca posmatranja $\overrightarrow{V}$, razmatra se vrednost polovine ugla između pravca posmatranja $\overrightarrow{V}$ i pravca svetla $\overrightarrow{L}$ i spekularna komponenta osvetljenja se računa po formuli:

$$I_{spec} = I_p k_s (\overrightarrow{N} \cdot \overrightarrow{H})^s$$

gde je sa $\overrightarrow{H}$ označen normalizovan vektor koji polovi ugao između vektora $\overrightarrow{V}$ i $\overrightarrow{L}$ (slika 8.22) i može se izračunati po sledećoj formuli:

$$\overrightarrow{H} = \frac{\overrightarrow{L} + \overrightarrow{V}}{|\overrightarrow{L} + \overrightarrow{V}|}$$



Slika 8.22: Blin-Fongov model osvetljenja.

Blin-Fongov model osvetljenja je efikasniji od Fongovog modela u slučaju kada su posmatrač i izvor svetla dovoljno daleko. Ovo odgovara situaciji kada imamo direkcioni izvor svetla i kameru koja odgovara paralelnom projektovanju. Naime, u ovom slučaju, vektori $\overrightarrow{L}$ i $\overrightarrow{V}$ imaju konstantnu vrednost, pa i vektor $\overrightarrow{H}$ ima konstantnu vrednost, odnosno ne zavisi od pozicije i zakrivljenosti površi. Stoga se vektor $\overrightarrow{H}$ može izračunati jednom za svaki izvor svetla i onda koristiti za ceo frejm ili sve dok izvor svetla i posmatrač ostaju u istom relativnom položaju. Ovo nije slučaj sa Fongovim modelom osvetljenja kod koga vektor refleksije zavisi od zakrivljenosti površi, tj. od vektora $\overrightarrow{N}$ i mora se iznova računati za svaki piksel slike.

8.3.7 Hromatska svetlost

Prethodno razmatrani modeli i odgovarajuće jednačine razmatrali su samo monohromatsku svetlost. Hromatska svetlost se obično obrađuje tako što se posebno računaju vrednosti njenih komponenti.

Neka je sa (O_{dR}, O_{dG}, O_{dB}) zadata difuzna crvena, zelena i plava komponenta nekog objekta u RGB kolor sistemu. U ovom slučaju se tri primarne komponente osvetljenja I_{pR}, I_{pG}, I_{pB} reflektuju proporcionalno sa vrednostima $k_d O_{dR}$, $k_d O_{dG}$ i $k_d O_{dB}$. Stoga, u RBG kolor modelu jednačina:

$$I = I_a k_a + f_{att} I_p (k_d (\overrightarrow{L} \cdot \overrightarrow{N}) + k_s (\overrightarrow{R} \cdot \overrightarrow{V})^s)$$

daje odgovarajuću jednačinu za crvenu komponentu:

$$I_R = I_{aR} k_a O_{dR} + f_{att} I_{pR} (k_d O_{dR} (\overrightarrow{L} \cdot \overrightarrow{N}) + k_s (\overrightarrow{R} \cdot \overrightarrow{V})^s)$$

Slično bi se zapisale i jednačine za zelenu i plavu komponentu.

Prethodno razmatranje podrazumeva da se interakcija svetlosti sa objektom može u potpunosti modelovati RGB kolor modelom. U idealnom slučaju, hromatsko osvetljenje bi trebalo računati kombinovanjem (neprekidnih) rezultata za sve boje iz spektra. Na primer, u zavisnosti od talasne dužine svetlosti Fongova jednačina osvetljenosti dobija finiji oblik:

$$I_\lambda = I_{a\lambda} k_a O_{d\lambda} + f_{att} I_{p\lambda} (k_d O_{d\lambda} (\overrightarrow{L} \cdot \overrightarrow{N}) + k_s O_{s\lambda} (\overrightarrow{R} \cdot \overrightarrow{V})^s)$$

gde je $O_{s\lambda}$ *spekularna boja objekta*, odnosno reflektovana boja sjajnog dela objekta. No, i opisani pojednostavljeni model obično daje prihvatljive rezultate.

8.4 Modeli senčenja

Kao što smo već pomenuli, problem senčenja se odnosi na proces bojenja kompletnog objekta, površi, odnosno poligona u trodimenzionoj sceni. Svaka površ se može senčiti izračunavanjem vektora normale površi u svakoj tački koja je vidljiva i primenom odgovarajućeg modela osvetljenja u toj tački. Međutim, ovakav algoritam bio bi jako skup, te se zbog toga razmatraju alternativni algoritmi senčenja.

8.4.1 Ravansko senčenje

Najjednostavniji model senčenja jeste *ravansko (konstantno) senčenje* (eng. flat shading) kod koga se čitav poligon senči istom bojom. U ovom modelu senčenja model osvetljenja se poziva samo jednom po poligonu da bi se izračunala vrednost intenziteta kojom se taj poligon senči. Dakle, u ovom modelu će čitav poligon biti obojen istom bojom. Ovaj model daje prihvatljiv prikaz ukoliko su zadovoljene sledeće pretpostavke:

- izvor svetla je beskonačno daleka tačka, pa je vrednost $\overrightarrow{L} \cdot \overrightarrow{N}$ konstantna za sve tačke jednog poligona,
- tačka posmatranja je beskonačno daleka tačka, pa je vrednost $\overrightarrow{V} \cdot \overrightarrow{N}$ konstantna za sve tačke jednog poligona,
- poligoni odgovaraju stvarnom objektu i nisu njegova aproksimacija.

Ako su vrednosti $\overrightarrow{L}$ i $\overrightarrow{N}$ konstantne, konstantna je i vrednost $\overrightarrow{R}$ (jer je vektor $\overrightarrow{R}$ simetričan vektoru $\overrightarrow{L}$ u odnosu na $\overrightarrow{N}$). Dakle, pod ovim pretpostavkama je u Fongovom modelu vrednost boje svih tačaka

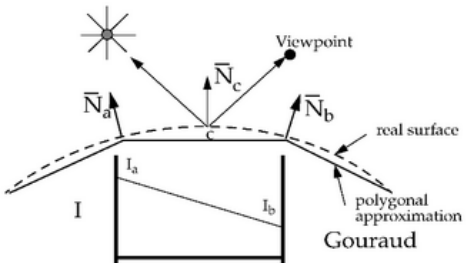
jednog poligona ista te ovaj model senčenja za svaku tačku poligona daje tačne vrednosti. Poslednja pretpostavka, da mreža poligona odgovara stvarnom objektu najčešće nije tačna, što negativno utiče na rezultujuću sliku (slika 8.23 levo).

8.4.2 Interpolirano senčenje

Umesto ravanskog, može se koristiti *interpolirano (Guroovo) senčenje* (eng. Gouraud shading) u kome se za svako teme mreže poligona računa intenzitet svetla, dok se intenzitet svetlosti u pojedinačnoj tački poligona izračunava linearnom interpolacijom vrednosti intenziteta svetlosti izračunatim u njegovim temenima (slika 8.23 desno). Dakle, ako je tačka poligona bliža nekom temenu od ostalih, uticaj tog temena na njen intenzitet svetlosti biće veći. Za izračunavanje vrednosti intenziteta u temenu potreban nam je vektor normale, međutim jedno teme može pripadati većem broju strana, pa se vektor normale u temenu računa uprosečavanjem vektora normala svih poligona kojima pripada to teme i nakon toga se za svako teme primenjuje odgovarajući model osvetljenja.

Slika 8.23: Ilustracija ravanskog i interpoliranog senčenja na modelu čajnika.

Istaknimo to da se Guroovim senčenjem mogu propustiti oblasti istaknute spekularne refleksije koji se javljaju usred poligona jer se umesto računanja intenziteta u svakoj od tačaka vrši interpolacija vrednosti u temenima. Na slici 8.24 prikazan je slučaj kada je vrednost spekularne komponente u temenima a i b mala, dok je spekularna komponenta u tački c velika jer se pravac posmatranja poklapa sa odbojnim zrakom svetlosti, međutim ovaj efekat u slučaju Guroovog senčenja neće biti uočljiv jer se intenzitet svetlosti dobija interpoliranjem vrednosti u tačkama a i b.

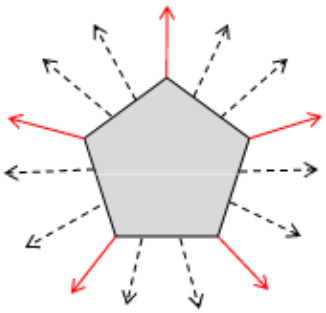


Slika 8.24: Računanje spekularne komponente osvetljenja u slučaju Guroovog senčenja.

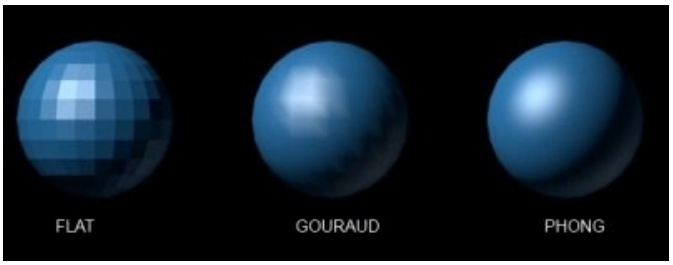
8.4.3 Fongovo senčenje

Fongovim senčenjem (eng. Phong shading) se vektor normale u proizvoljnoj tački poligona računa interpolacijom vektora normala u temenima, a zatim se vrednost intenziteta svetlosti u toj tački dobija na osnovu Fongovog modela osvetljenja. Ovim modelom se dobija preciznija informacija o stvarnoj vrednosti vektora normale u svakoj od tačaka (slika 8.25).

Fongov model senčenja daje bolju aproksimaciju od Guroovog modela senčenja u slučaju oblih i glatkih objekata (slika 8.26). Pored toga, s obzirom na to da se vrši interpolacija vektora normala temena poligona, Fongov model nema pomenuti problem da se spekularni efekat izgubi iz unutrašnjosti poligona. Međutim, Fongovo senčenje je računski mnogo zahtevnije od Guroovog senčenja jer se model osvetljenja primenjuje, umesto na svako teme, na svaki piksel kako bi se dobila vrednost intenziteta svetlosti.



Slika 8.25: Linearna interpolacija vektora normala u temenima.



Slika 8.26: Razlika između konstantnog, Guroovog i Fongovog modela senčenja.

8.5 Senke

Senke predstavljaju rezultat interakcije između svetlosti i objekata. One sceni daje realističnost: njima se dobija utisak dubine na sceni, lakše se uočava kakav je prostorni odnos objekata na sceni i njihova relativna dubina (rastojanje od posmatrača), kao i kako se objekti kreću.

Senke se mogu podeliliti na oštre i meke (slika 8.27) u zavisnosti od toga da li ih proizvodi tačkasti izvor svetla ili prošireni. U realnom životu skoro sve senke su meke, međutim, u situacijama kada se objekti posmatraju sa dovoljno velike udaljenosti senke mogu delovati oštro.

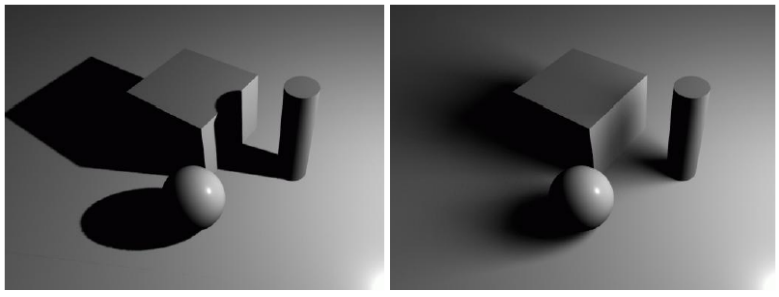


Figure 2(a): Hard Shadow Figure 2(b): Soft Shadow

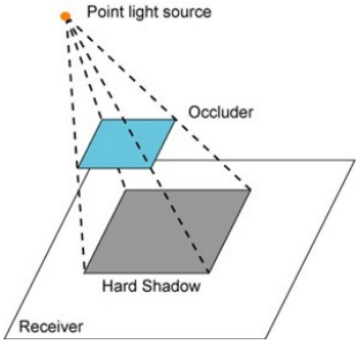
Slika 8.27: Razlika između oštrih i mekih senki (preuzeto sa http://www.msrblog.com/science/computer/real-time-soft-shadow-rendering.html).

Algoritmi za utvrđivanje vidljivosti određuju koji se delovi površi mogu videti iz tačke posmatranja, dok algoritmi za senke određuju koji se delovi površi mogu videti iz izvora svetla. Oni delovi površi koji se ne vide iz izvora svetla su u senci.

U situaciji kada na sceni postoji veći broj izvora svetla, npr. njih m, Fongova jednačina osvetljenosti postaje:

$$I_\lambda = I_{a\lambda} k_a O_{d\lambda} + \sum_{1 \le i \le m} S_i f_{att} I_{p\lambda} (k_d O_{d\lambda} (\overrightarrow{L} \cdot \overrightarrow{N}) + k_s O_{s\lambda} (\overrightarrow{R} \cdot \overrightarrow{V})^n)$$

gde S_i ima vrednost 0 ako je izvor svetla i blokiran u ovoj tački, a inače ima vrednost 1. Pritom, izvor svetla može biti blokiran nekim drugim objektom sa scene ili samim sobom. Primetimo da su i oblasti koje su u senci u odnosu na sve izvore svetla i dalje osvetljene ambijentalnom svetlošću.

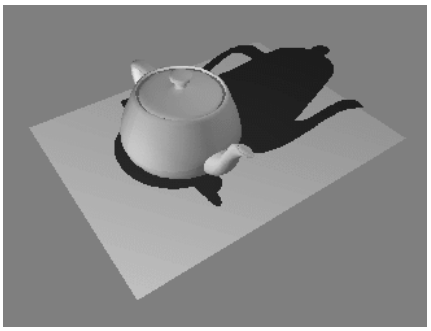


Slika 8.28: Računanje senki iz tačkastog izvora svetla.

Razmotrićemo nekoliko različitih pristupa računanju senki: projektivne senke, algoritam koji koristi bafer senki i algoritam zasnovan na zapreminama senki.

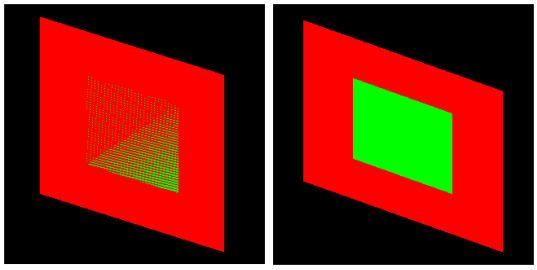
8.5.1 Projektivne senke

U slučaju tačkastog izvora svetla, senke je moguće generisati i bez izvršavanja bilo kakvih testova vidljivosti. Naime, senka odgovara perspektivnoj projekciji poligona na površ osnove sa centrom projekcije u izvoru svetla (slika 8.28). Dakle, sceni je moguće dodati senke tako što ih crtamo kao nezavisne, ravne i tamne objekte. Ova vrsta senki naziva se *projektivna senka* (eng. projective shadow) i pogodna je za složene objekte koji bacaju senke na jednostavne površi. U ovom pristupu svaki objekat se renderuje dva puta: u prvom prolazu vrši se uobičajeno renderovanje objekata na sceni sa uključenim bojama i teksturama predviđenim za te objekte, dok se u drugom prolazu isključuju osvetljenje i teksture, objekat se projektuje iz izvora svetla na ravan osnove i renderuje u potpunosti crnom bojom, odnosno bojom senke (slika 8.29). Ovo je jednostavno uraditi ako se vrši projekcija na beskonačnu ravnu površ, međutim, postaje znatno teži geometrijski problem ako je površ na koju se vrši projekcija složena (npr. projekcija na stepenice). Naime, projekciju je u tom slučaju potrebno odseći u odnosu na površ osnove. To je moguće uraditi korišćenjem stensil bafera, kojim se maksiraju delovi slike, tako da se senke iscrtavaju samo na pozicijama gde je postavljena maska. O samom stensil baferu biće više reči u daljem tekstu.

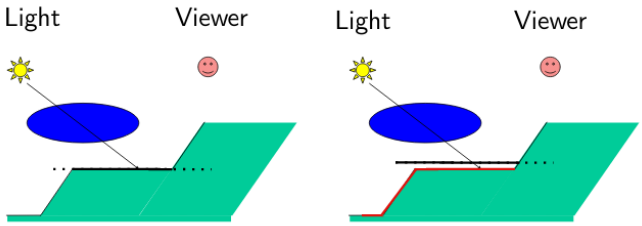


Slika 8.29: Projektivna senka.

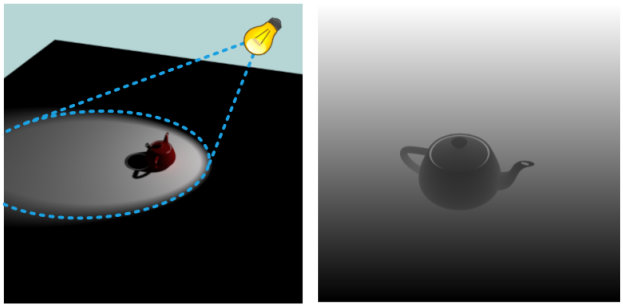
Kod projektivnih senki može se javiti problem s obzirom na to da projekcije objekata imaju istu vrednost dubine kao i ravan na koju se vrši projektovanje (slika 8.30). Naime, s obzirom da senka i objekat na koji se vrši projekcija imaju isto rastojanje do kamere, može doći do z-konflikta. U ovoj situaciji skoro je slučajno koja će od tih poligona biti iscrtan na poziciji nekog piksela. Posebno je nezgodan moment kada se scena ili kamera pomeraju, jer se može dogoditi da u jednom momentu bude iscrtan piksel jednog poligona, nakon toga na poziciji istog piksela drugi poligon, pa ponovo prvi itd. Jedan od načina za rešavanje ovog problema jeste da se poligoni koji čine senku neznatno izdignu po z-koordinati, odnosno da se približe izvoru svetla: na taj način geometrija senke biće iscrtana iznad površi na koju se vrši projekcija (slika 8.31). Još neki problemi koji se mogu javiti kod projektivnih senki su da rezultujuća senka ima oštre ivice i da, ako postoji tekstura na ravni osnove, senke mogu imati loš izgled.



Slika 8.30: Ilustracija z-konflikta (preuzeto sa wikipedia).



Slika 8.31: Rešavanje z-konflikta približavanjem senki izvoru svetla (preuzeto sa http://www.inf.ed.ac.uk/teaching/courses/cg/lectures/slides11.pdf).



Slika 8.32: Pogled iz pozicije kamere (levo) i iz izvora svetla (desno). Tamnije vrednosti na ilustraciji bafera senki odgovaraju bližim tačkama (preuzeto sa http://www.downloads.redway3d.com/downloads/public/documentation/bk_re_shadow_mapping_detailed.html).

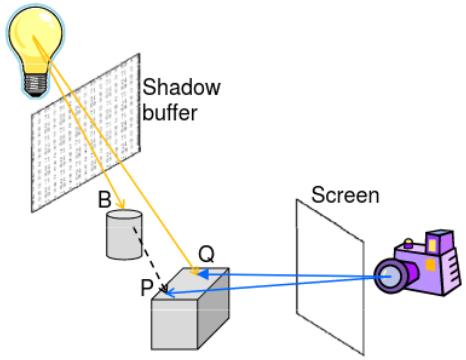
8.5.2 Algoritam za senke zasnovan na baferu senki

Bafer senki (eng. shadow buffer) je bafer dubine renderovan umesto iz pozicije kamere iz pozicije izvora svetla. On se često naziva i *mapom senki* (eng. shadow map). Svaki element bafera senki sadrži rastojanje do prvog preseka zraka sa početkom u izvoru svetla kroz odgovarajući piksel sa nekim objektom sa scene (slika 8.32). Kasnije, kada želimo da utvrdimo da li je neka tačka P u trodimenzionom prostoru osvetljena nekim izvorom svetlosti, određujemo perspektivnu projekciju tačke P iz izvora svetlosti na mapu senki i vršimo proveru da li je ona dalja od izvora svetlosti nego što je vrednost rastojanja koja se čuva na odgovarajućem mestu u mapi. Ako jeste, onda je ona zaklonjena nekim bližim objektom i stoga nije osvetljena tim izvorom svetla (slika 8.33). Na ovaj način moguće je slici dodati senke.

Mapa senki se može popuniti u jednom prolazu kroz geometriju scene pre nego što se razmotri vidljivost iz pozicije kamere. Kada se tokom procesa renderovanja scene iz pozicije kamere pojavi potreba za izračunavanjem senki, koristimo informacije sačuvane u mapi senki: tačka se renderuje tako da nije u senci ako je vidljiva i iz pozicije kamere i iz izvora svetla. Ovaj proces poznat je i pod nazivom *dvoprolazni z-bafer algoritam* (eng. two pass z-buffer algorithm).

Nedostatak ovog pristupa jeste u tome da kada je kamera bliža sceni nego izvor svetla, onda mnogi pikseli ekrana mogu biti pokriveni samo jednim pikselom mape senki. To što se jedna vrednost piksela sa mape senki koristi za utvrđivanje statusa svih tačaka koje se projektuju u taj piksel može voditi nekim nepoželjnim aliasing efektima (slika 8.34).

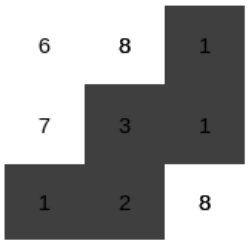
Korišćenje mapi senki daje binarni rezultat: objekat je u odnosu na izvor svetla u senci ili nije, odnosno doprinos nekog izvora svetla jednak je 0 ili 1. Senke dobijene na ovaj način imaju oštre ivice. Umesto toga, moguće je simulirati meke senke testiranjem vrednosti rastojanja objekta čiju osvetljenost



Slika 8.33: Korišćenje bafera senki za utvrđivanje koje tačke su u senci: tačka P je u senci, a tačka Q nije (preuzeto sa https://www.cs.auckland.ac.nz/compsci373s1c/ChristofsLectures/CS373-Part1-Lecture08-Illumination2-1pp.pdf).

Slika 8.34: Efekat nazubljenosti dobijen naivnim preslikavanjem senki (preuzeto sa http://www-sop.inria.fr/reves/Marc.Stamminger/psm/).

razmatramo od izvora svetla i u odnosu na susedne piksele i izračunati prosek dobijenih vrednosti. Razmotrimo primer dela mape senki sa slike 8.35 koji predstavlja susede piksela koji trenutno razmatramo. Neka je vrednost rastojanja objekta od izvora svetla jednaka 5: vidimo da je za četiri od devet susednih piksela vrednost u mapi veća od date, te je doprinos ovog izvora svetla datom pikselu jednaka $\frac{4}{9}$.

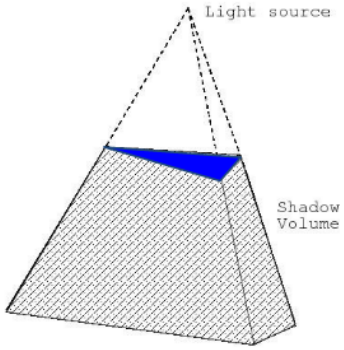


Slika 8.35: Primer dela mape senki koji odgovara susedima razmatranog piksela (preuzeto sa slajdova Univerziteta Brown).

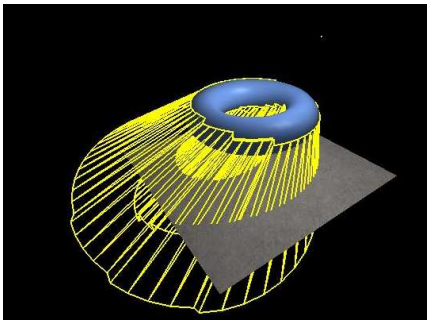
8.5.3 Algoritam za senke zasnovan na zapreminama senki

U realnosti, senka koju objekat baca čini trodimenzionalnu zapreminu, a ne dvodimenzionu oblast u ravni. Za svaki par izvora svetlosti i objekta može se izračunati oblast u kojoj objekat zaklanja svetlost i ona se naziva *zapreminom senke* (eng. shadow volume). Najpre je potrebno izračunati siluetu objekta iz perspektive izvora svetla: ona razdvaja deo objekta koji je vidljiv iz izvora svetlosti od onog koji nije. Silueta objekta sadrži ivice objekta koje dele dva trougla od kojih je jedan okrenut izvoru svetla, a drugi nije. Nakon toga se dobijena silueta projektuje duž zraka svetlosti i generišu se poligoni koji povezuju siluetu sa njenom projekcijom: na taj način dobija se zapremina senke. Kao što je prikazano na slici 8.36

za svaku ivicu siluete objekta postoji jedan četvorougao kojim se ograničava zapremina senke.

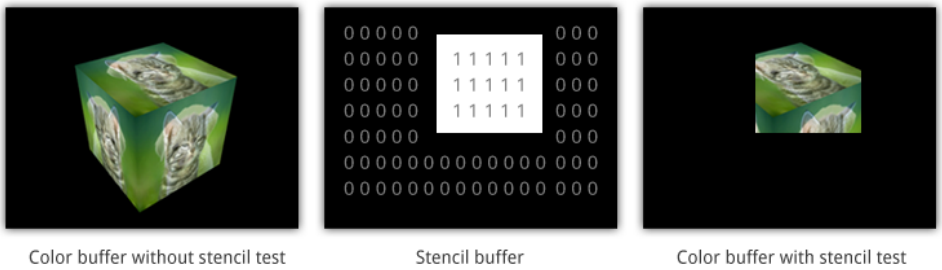


Slika 8.36: Zapremina senke definisana izvorom svetla i objektom (preuzeto iz http://www.inf.ed.ac.uk/teaching/courses/cg/lectures/slides11.pdf.



Slika 8.37: Zapremina senke (preuzeto sa www.ozone3d.net/tutorials/stencil_shadow_volumes.php).

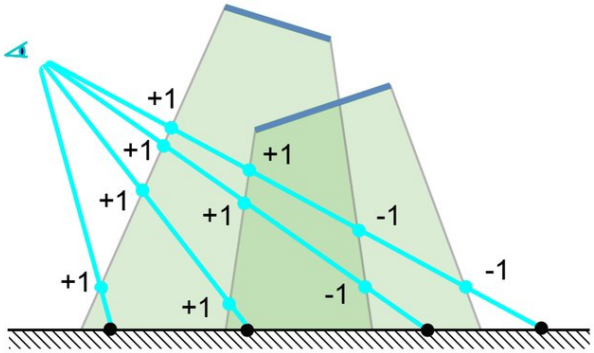
Zapremina senke ograničena je skupom *poligona senke* (eng. shadow polygon). Poligoni senke se ne renderuju, već se koriste za utvrđivanje da li su drugi objekti u senci. Naime, tačka P je u senci u odnosu na izvor svetla L ako neka od zapremina senki V izračunata za izvor svetla L sadrži P (slika 8.37). Ovo se može brzo utvrditi korišćenjem većeg broja prolaza i *stensil bafera*, odnosno *bafera šablona* (eng. stencil buffer). Stensil bafer predstavlja bafer u memoriji na grafičkoj kartici koji se koristi za različite specijalne efekte i najčešće sadrži 8 bita informacije po pikselu. Svakom pikselu u stensil baferu dodeljuje se neka vrednost i te vrednosti se često koriste da maskiraju delove slike tokom obrade kao što je to slučaj sa šablonima tokom crtanja na papiru. Na slici 8.38 dat je primer primene stensil bafera za izdvajanje jednog dela slike.



Slika 8.38: Primena stensil bafera za izdvajanje dela slike.

Sam algoritam za određivanje senki se izvršava na sledeći način: najpre se za svaki objekat na sceni određuje njegova zapremina senke, a zatim se scena renderuje korišćenjem samo ambijentalnog svetla. U drugom prolazu anulira se sadržaj stensil bafera, a zatim vrši renderovanje zapremine senki tako da pikseli trouglova sa prednje strane, odnosno zadnje strane inkrementiraju, odnosno dekrementiraju redom vrednosti u stensil baferu (slika 8.39). Nakon faze renderovanja, stensil bafer klasifikuje tačke na one u senci i one koje to nisu: ako je vrednost u stensil baferu veća od nule, to znači da je tačka unutar

zapremine senke, inače je van. Ovo tvrđenje se zasniva na činjenici da je tačka P u senci ako i samo ako duž datog zraka postoji veći broj preseka kojima se ulazi u zapreminu senke, nego izlaznih preseka. U trećem prolazu, scena se renderuje korišćenjem punog modela osvetljenja i vrednosti u stensil baferu se koriste da maskiraju piksele u senci. Za razliku od mapa senki, algoritam zasnovan na zapreminama senki ne pati od aliasing problema. Nedostatak ovog algoritma je taj što zapremina senki često pokriva veliki broj piksela.

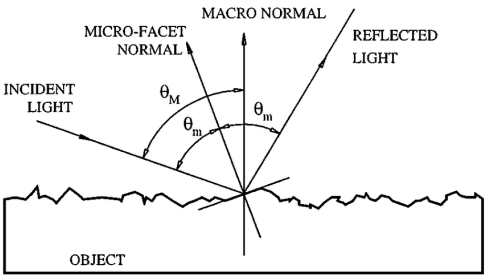


Slika 8.39: Algoritam zasnovan na zapreminama senki (preuzeto iz master teze "Accurate Soft Shadows in Real-Time Applications", Michael Schwarlzler).

8.6 Fizički zasnovani modeli osvetljenja

Tokom vremena razvijani su i složeniji modeli osvetljenja. Neki od njih su Kuk-Torensov model (Robert Cook, Kenneth Torrance) i Oren-Najarov model (Michael Oren, Shree Nayar) i oni su za razliku od prethodna dva modela zasnovani na pravilima fizike, tačnije optike. U poređenju sa prethodno razmatranim Fongovim i Blin-Fongovim modelom osvetljenja, dobijeni prikaz je realističniji.

Fizički zasnovano renderovanje (eng. physically based rendering, skraćeno PBR) je pristup renderovanju slika na način koji modeluje interakciju svetala sa površima korišćenjem pravila optike. Kreiran je od strane kompanije Dizni za potrebe dugometražnih animacija.



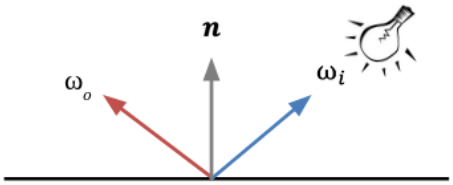
Slika 8.40: Ilustracija mikropovrši.

Oni se zasnivaju na konceptu *mikropovrši* (eng. microfacet): pretpostavljamo da se površi koje nisu savršeno glatke sastoje od veoma malih površi, poređanih na pogodan način, od kojih je svaki savršeno reflektujuće ogledalo (slika 8.40). Mikropovrši imaju normale koje su nekako raspodeljene oko normale glatke površi koju aproksimiraju. Stepen odstupanja normala vektora mikropovrši od vektora normale glatke površi određen je tim koliko je površ gruba.

Za fizički zasnovane modele osvetljenja karakteristično je to što čuvaju energiju: naime, izlazna energija svetlosti ne može da prevaziđe ulaznu svetlosnu energiju, naravno osim kod površi koje emituju svetlost. Naime, reflektovanje više svetlosti od toga koliko je proizvedeno nema fizički smisla jer se energija ne može stvoriti ni iz čega. Dakle, kada svetlost pogodi objekat, energija te svetlosti se mora očuvati: može se desiti reflektovanje, apsorbovanje, razbacivanje ili konvertovati u toplotnu energiju.

U ovim modelima se materijal od koga je objekat zadaje *dvosmernom funkcijom raspodele refleksije* (eng. bi-directional reflectance distribution function, skraćeno BRDF) f_r, kojom se za dati upadni zrak svetlosti ω_i i datu normalu površi n zadaje procenat svetlosti koji se reflektuje duž proizvoljnog smera ω_o (slika 8.41):

$$f_r(\omega_i, \omega_o) : \mathbb{R}^3 \times \mathbb{R}^3 \to [0, 1]$$

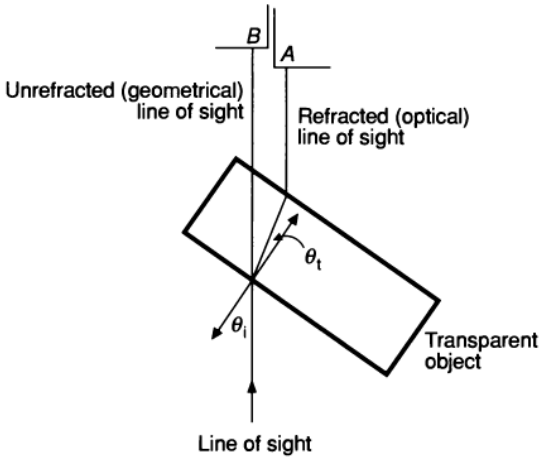


Slika 8.41: BRDF funkcija.

Pritom, što je površ refleksivnija, to je vrednost funkcije f_r veća. Ovakvi modeli osvetljenja koriste fizički zasnovanu funkciju BRDF, koja se pridržava principa očuvanja energije.

8.7 Transparentnost

Neki objekti, odnosno materijali kao što je recimo staklo su transparentni, tj. propuštaju deo svetlosti. Obično se ta svetlost *prelama* (eng. refract), odnosno svetlosni zraci se lome prilikom prolaska kroz tu površ, ali se taj efekat često zanemaruje kako bi se računanje pojednostavilo. U nastavku teksta zanemarićemo efekat prelamanja svetlosti i podrazumevaćemo scenario po kome šta god je vidljivo duž linije pogleda kroz transparentnu površ se geometrijski i nalazi na toj liniji pogleda (slika 8.42).



Slika 8.42: Ako se u obzir uzme i prelamanje svetlosti, objekat A je vidljiv kroz transparentni objekat duž linije pogleda; ako se prelamanje svetlosti ignoriše, duž linije pogleda vidljiv je objekat B.

Razmotrićemo dva modela transparentnosti koja ne vode računa o prelamanju svetlosti, a koja se često koriste da aproksimiraju način na koji se boje dva objekta kombinuju kada se jedan objekat vidi kroz drugi. To su *interpolirana transparentnost* (eng. interpolated transparency) i *filtrirana transparentnost* (eng. filtered transparency).

Kod intepolirane transparentnosti ukoliko je poligon P_1 transparentan i nalazi se ispred neprozirnog poligona P_2, intenzitet svetlosti za piksel u preseku projekcija dva poligona dobija se linearnom interpolacijom pojedinačnih osvetljenosti izračunatih za dva data poligona:

$$I_\lambda = (1 - k_{t_1})I'_\lambda + k_{t_1}I''_\lambda$$

gde je I'_λ intenzitet svetlosti za poligon P_1, I''_λ intenzitet svetlosti za poligon P_2, a k_{t_1} je *koeficijent transmisije* (eng. transmission coefficient) za poligon P_1 i on predstavlja meru transparentnosti poligona P_1. Vrednost k_{t_1} je između 0 i 1: kada je jednaka 0, poligon je neproziran, a kada je jednaka 1 poligon je potpuno transparentan.

Kod filtrirane transparentnosti poligon se tretira kao filter koji selektivno propušta različite talasne dužine i ovaj model transparentnosti se može opisati jednačinom:

$$I_\lambda = I'_\lambda + k_{t_1}O_{t\lambda}I''_\lambda$$

gde je $O_{t\lambda}$ *boja transparentnosti* (eng. transparency color) poligona P_1 i može biti posebno zadavana za svaku vrednost talasne dužine λ. Primetimo da ako je koeficijent transmisije poligona P_1 jednak 0, poligon P_2 neće imati uticaj na osvetljenost datog piksela.

U oba pomenuta modela transparentnosti, ako se još neki transparentni poligoni nalaze ispred ovih poligona, onda se izračunavanje rekurzivno poziva za poligone u redosledu od najudaljenijeg do najbližeg, pri čemu se svaki put koristi prethodno izračunato I_λ kao I''_λ.

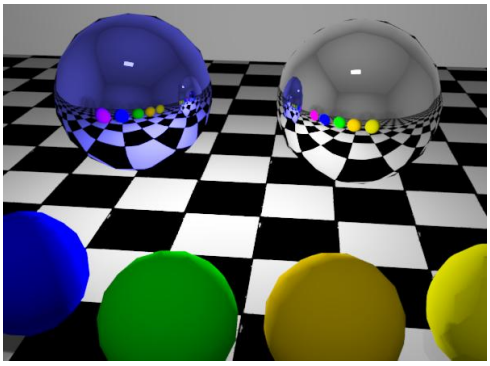
Dosta teže je modelovati transparentnost koja uključuje i prelamanje svetlosti. Prelamanje svetlosti može se opisati *Snelovim zakonom*:

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{\nu_1}{\nu_2}$$

gde su θ_1 i θ_2 redom upadni ugao svetlosti i ugao prelamanja svetlosti, a ν_1 i ν_2 *indeksi prelamanja materijala* (eng. indices of refraction) kroz koje svetlost prolazi. Indeks prelamanja materijala jednak je količniku brzine svetlosti u vakuumu i brzine svetlosti za dati materijal. Indeks prelamanja vakuuma jednak je 1, dok je za ostale materijale veći od 1.

8.8 Međuobjektne refleksije i globalno osvetljenje

Međuobjektne refleksije se javljaju kada se na površini objekta reflektuju druge površine u okruženju (slika 8.43). Ovaj efekat može da se kreće od spekularne refleksije koja se menja sa položajem posmatrača), pa do difuzne refleksije (na koju ne utiče pozicija posmatrača).

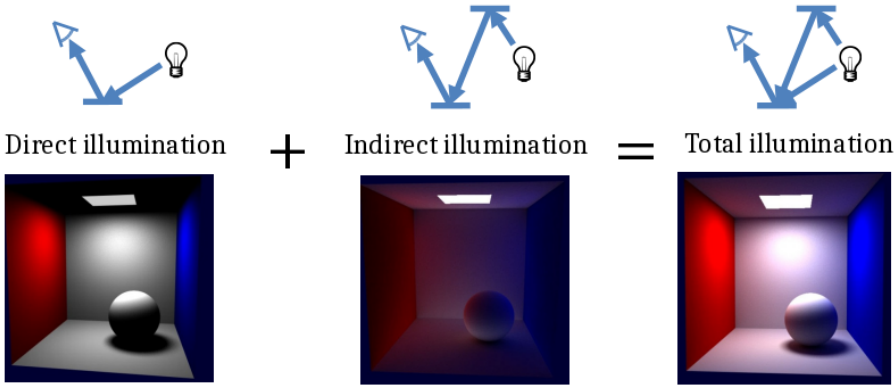


Slika 8.43: Međuobjektne refleksije.

Modeli osvetljenja računaju boju u nekoj tački u terminima svetlosti direktno emitovane od strane izvora svetla i terminima svetlosti koja dolazi do tačke nakon reflektovanja i prolaska (prelamanja) kroz razne površi. Ova indirektno reflektovana i transmitovana svetlost se obično zove *indirektno osvetljenje* (eng. indirect illumination), dok se pod *direktnim osvetljenjem* (eng. direct, local illumination) podrazumeva svetlost koja dolazi direktno iz izvora svetla do tačke koja se senči. I Fongov i Blin-Fongov model osvetljenja razmatraju samo direktno osvetljenje. Kombinovanjem direktnog i indirektnog osvetljenja, dobija se *globalno osvetljenje* (eng. global illumination), koje uzima u obzir interakciju svetlosti sa svih površi na sceni (slika 8.44). Jedan od metoda za postizanje globalnog osvetljenja na sceni jeste rekurzivni rej trejsing algoritam.

8.9 Rekurzivni rej trejsing algoritam

Rekurzivni rej trejsing algoritam rešava problem vidljivosti tako što u pravcu datog zraka određuje tačku preseka koja je najbliža posmatraču (i njenu boju). Da bismo odredili senke, upućujemo dodatni zrak iz tačke preseka do svakog izvora svetla. Ako neki od ovih zraka preseca neki netransparentni objekat duž svog puta, onda je polazna tačka u senci i taj izvor svetla ne doprinosi njenoj boji. Pored toga, iz tačke preseka se upućuju i reflektujući (odbojni) zrak i zrak dobijen nakon prelamanja. Ovi zraci kao i zrak ka izvoru svetla nazivaju se *sekundarni zraci* (eng. secondary ray), dok se zrak iz pozicije kamere naziva *primarni zrak* (eng. primary ray). U nastavku teksta je prikazan pseudokod rekurzivnog rej trejsing algoritma. Procedura `RT_trace` određuje najbliži presek zraka sa objektima na sceni i poziva proceduru `RT_shade` da odredi boju u toj tački. Ona najpre određuje ambijentalnu boju objekta u presečnoj tački, zatim upućuje po jedan zrak do svakog izvora svetla kako bi odredio njegov doprinos boji objekta: neprozirni objekti blokiraju svetlost u potpunosti, dok se kod transparentnih vrši skaliranje doprinosa



Slika 8.44: Globalno osvetljenje.

tog izvora svetla. Ako dubina rekurzije nije prevelika, onda se rekurzivni pozivi bave obradom zraka refleksije (kod reflektivnih objekata) i zraka prelamanja (za transparentne objekte).

```
select center of projection and window on view plane;
for each scan line in image do
  for each pixel in scan line do
    begin
      determine ray from center of projection through pixel;
      pixel := RT_trace(ray, 1);
    end

{Intersect ray with objects and compute shade at closest intersection.}
{Depth is current depth in ray tree.}
procedure RT_trace(ray: RT_ray; depth: integer) : RT_color;
begin
  determine closest intersection of ray with an object;

  if object hit then
    begin
      compute normal at intersection;
      RT_trace := RT_shade (closest object hit, ray,
                            intersection, normal, depth);
    end
  else
    RT_trace := BACKGROUND_VALUE;
end;

{Compute shade at point on object, tracing rays for shadows,
reflection, refraction}
procedure RT_shade (
  object : RT_object; {Object intersected}
  ray : RT_ray;       {Incident ray}
  point : RT_point;   {Point of intersection to shade}
  normal : RT_normal; {Normal at point}
  depth : integer     {Depth in ray tree}
) : RT_color;
var
  color : RT_color    {Color of ray}
  rRay, tRay, sRay : RT_ray; {Reflected, refracted, and shadow rays}
  rColor, tColor : RT_Color; {Reflected and refracted ray colors}

begin
  color := ambient term;
  for each light do
```

```
    begin
      sRay := ray to light from point;
      if dot product of normal and direction to light is positive then
        compute how much light is blocked by opaque and transparent
        surfaces, and use to scale diffuse and specular terms before
        adding them to color;
    end

 if (depth < maxDepth) then   {Return if depth too deep.}
   begin
     if object is reflective then
       begin
         rRay := ray in reflection direction from point;
         rColor := RT_trace(rRay, depth+1);
         scale rColor by specular coefficient and add to color;
       end

    if object is transparent then
      begin
        tRay := ray in refraction direction from point;
        if total internal reflection does not occur then
          begin
            tColor := RT_trace(tRay, depth+1);
            scale tColor by transmission coefficient and add to color;
          end
      end
    end
  RT_shade := color; {Return color of ray}
end.
```

8.10 Pitanja

8.1 Šta je zadatak problema osvetljenja, a šta problema senčenja?

8.2 Koje je sve parametre potrebno zadati za jedan izvor svetla?

8.3 Kakvo svetlo nazivamo ambijentalnim? Kako izgledaju objekti osvetljeni samo ambijentalnim svetlom?

8.4 Od čega sve zavisi osvetljenost tačke objekta tačkastim izvorom svetla?

8.5 Navesti Lambertovo kosinusno pravilo.

8.6 Koji izvor svetla nazivamo direkcionim?

8.7 Na koji način najčešće modulujemo faktor slabljenja izvora svetla f_{att}?

8.8 Da li vrednost faktora slabljenja izvora svetla može biti manja od 1?

8.9 Kakvu refleksiju nazivamo spekularnom, a kakvu difuznom?

8.10 Na koji način se moduluje spekularna komponenta osvetljenja u Fongovom, a na koji način u Blin-Fongovom modelu osvetljenja?

8.11 Da li su Fongov i Blin-Fongov model osvetljenja fizički zasnovani?

8.12 Koja su tri osnovna metoda senčenja?

8.13 Koji je nedostatak ravanskog senčenja? U kojoj situaciji je rezultat dobijen ravanskim senčenjem vizuelno prihvatljiv?

8.14 Kako se vrši senčenje polinoma kod Guroovog senčenja?

8.15 Koji je nedostatak Guroovog senčenja?

8.16 Na koji način se vrši senčenje kod Fongovog senčenja?

8.17 Šta se određuje algoritmima za senke?

8.18 Navesti tri osnovna načina za određivanje senki.

8.19 Opisati šta označava z-konflikt i na koji način se može rešiti.

8.20 Šta je bafer senki? U koje svrhe se koristi?

8.21 Kako se za neki objekat računa njegova zapremina senki?

8.22 Šta je stensil bafer? Koja mu je osnovna primena?

8.23 Navesti dva osnovna metoda za modelovanje transparentnosti.

8.24 Kakvo osvetljenje nazivamo indirektnim?

8.25 Opisati kako funkcioniše rekurzivni rej trejsing algoritam.

Glava 9

Boje

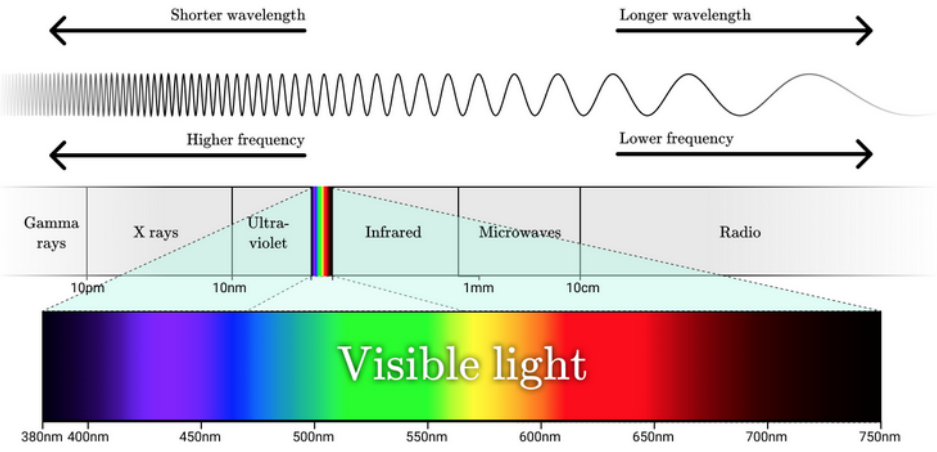
Veoma važan i moćan aspekt našeg doživljaja sveta koji nas okružuje se prenosi bojama. Ona predstavlja i važnu dimenziju vizuelne komunikacije u računarskoj grafici: kada se koristi na pravilan način ona može u velikoj meri da poboljša efikasnost poruke koju želimo da prenesemo.

Opažena boja nekog objekta zavisi od različitih faktora: od izvora svetla koji osvetljavaju objekat i njegovih karakteristika poput intenziteta, nijanse, pozicije i tipa izvora svetla, od geometrije samog objekta i svojstva materijala od kojeg je napravljen, od boje i svojstava okoline, ali i od samog čovekovog vizualnog sistema. Da bismo bolje razumeli zapis i predstavljanje boja u računarskoj grafici potrebno je razumeti osnove čovekovog vizualnog sistema koji je, možemo reći, sastavni deo grafičke protočne obrade.

Fenomen opažanja boja možemo okarakterisati kao subjektivno iskustvo koje naš mozak stvara prilikom interpretiranja fizičkog sveta i zavisi od naših prethodnih iskustava i asocijacija. Većina ljudi svetlost talasne dužine od oko 400nm opaža kao plavu, dok svetlost talasne dužine oko 700nm registruje kao crvenu. Vizualne senzacije uzrokovane obojenom svetlošću mnogo su bogatije nego one uzrokovane ahromatskom.

9.1 Vidljiva svetlost

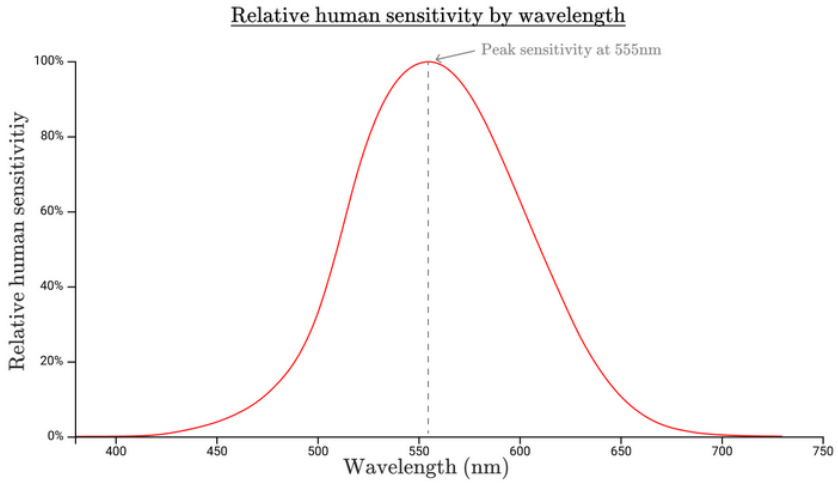
Radio talasi, infracrveno zračenje, ultraljubičasto, x-zračenje i vidljiva svetlost pokrivaju različite opsege talasnih dužina spektra elektromagnetnog zračenja (slika 9.1). Ljudskom oku vidljiva svetlost ima talasnu dužinu od oko 380 do oko 750 nanometara i predstavlja jako mali deo kompletnog spektra.



Slika 9.1: Spektar elektromagnetnog zračenja.

Svaka talasna dužina iz spektra vidljive svetlosti odgovara *monohromatskoj svetlosti* (eng. monochromatic light), tj. svetlosti jedne talasne dužine iz vidljivog dela spektra, odnosno uskog opsega talasnih dužina. Monohromatske boje poznate su i pod nazivom *spektralne boje* (eng. spectral colors). Istraživanja su pokazala da čovekovo oko nije jednako osetljivo na svetlost svih talasnih dužina, već da je mnogo osetljivije na svetlost talasne dužine oko 550nm nego recimo od 650nm ili od 450nm (slika 9.2). Ovo

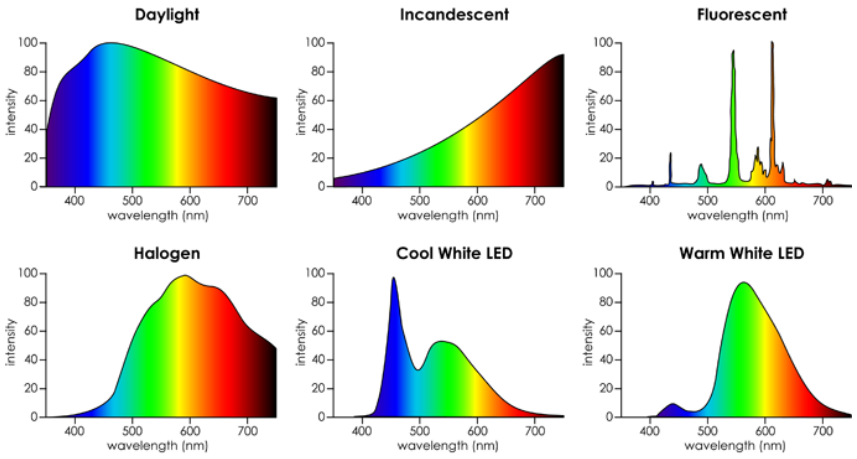
znači da ljudsko oko doživljava svetlost zelene boje kao sjajniju u odnosu na svetlost drugih talasnih dužina. Granice funkcije osetljivosti na svetlost različitih talasnih dužina definišu opseg vidljive svetlosti: talasne dužine van ovog opsega ne pripadaju vidljivoj svetlosti jer naše oko nije osetljivo na njih. Interesantno je da su oči životinja osetljive na drugačiji opseg talasnih dužina: ptice, na primer, mogu da vide ultraljubičasto zračenje u opsegu od 300nm do 400nm.



Slika 9.2: Osetljivost čovekovog oka na svetlost različite talasne dužine.

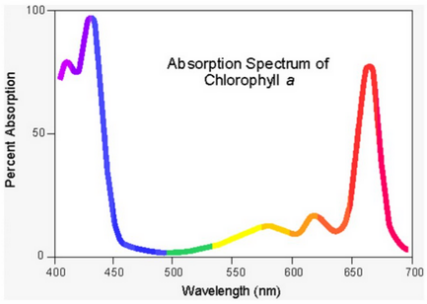
Boje koje su vidljive čoveku uglavnom predstavljaju neku težinsku kombinaciju monohromatske svetlosti.

Boje možemo da predstavimo preko emisionih spektara ili apsorpcionih spektara, u zavisnosti od toga da li se svetlost emituje ili apsorbuje. Emisioni spektar nastaje kada pobuđeni atomi ili molekuli oslobađaju energiju u obliku elektromagentnog zračenja, često u vidljivom delu sprektra. Drugim rečima, emisionim spektrom se za svaku talasnu dužinu zadaje koliko se svetlosti proizvelo (zagrevanjem, fuzijom ili na neki drugi način). Emisioni spektri su važni u kontekstu izvora svetla. Na slici 9.3 prikazani su emisioni spektri različitih izvora svetla. Primetimo da je dnevna svetlost gotovo uniformna mešavina svih frekvencija vidljive svetlosti, dok npr. kod halogene svetlosti to nije slučaj. Svaki izvor svetla ima svoj poseban emisioni spektar.



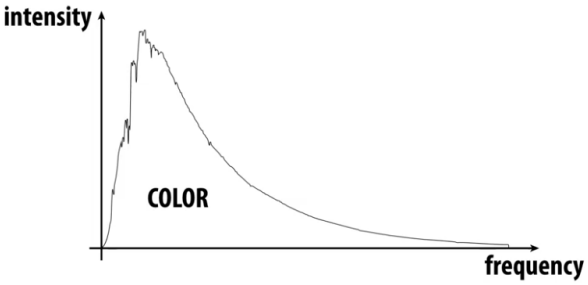
Slika 9.3: Emisioni spektri različitih izvora svetla.

S druge strane, apsorpcioni spektrom se za svaku talasnu dužinu zadaje procenat svetlosti te talasne dužine koji se apsorbuje ako se obasja belom svetlošću: ta svetlost se, na primer, može pretvoriti u toplotu. Apsorpcioni spektri su pogodni za opisivanje boje farbi i mastila. Na slici 9.4 prikazan je apsorpcioni spektar hlorofila, pri čemu je na koordinatnoj osi y zadat procenat svetlosti date talasne dužine koji se apsorbuje. Razmatranjem grafika možemo zaključiti da se apsorbuje gotovo sva plava i crvena boja, a nimalo zelene, te će biljke koje sadrže hlorofil pod dnevnim osvetljenjem reflektovati zelenu boju.



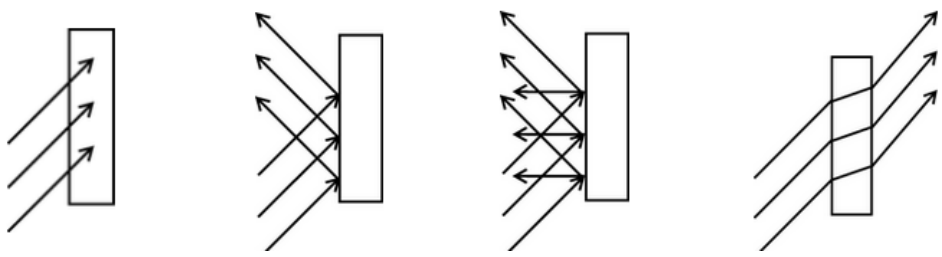
Slika 9.4: Apsorpcioni spektar hlorofila.

Dakle, ako želimo da precizno opišemo boju, onda je potrebno zadati intenzitet svetlosti ili procenat apsorpcije kao funkciju frekvencija (slika 9.5). Na ovaj način znamo sve sto bismo mogli da znamo o boji, dok je sve ostalo samo pogodna aproksimacija. Međtuim, jasno je da je u kontekstu računarske grafike potpuno nepraktično raditi sa spektrima boje, već je neophodno osmisliti način digitalnog kodiranja boja, poput modela boja RGB (eng. Red, Green, Blue), o čemu će u daljem tekstu biti više reči.



Slika 9.5: Opis boje kao funkcije frekvencija.

Ljudi koriste različite reči pri opisivanju boja objekata koji emituju svetlost i onih koji reflektuju svetlost. Kazaćemo da je neki objekat braon boje, ali gotovo nikada nećemo kazati da je svetlost braon boje. Istaknimo da boja objekta ne zavisi isključivo od samog objekta, već i od izvora svetla koji ga obasjava, od svetlosti koja pada na objekat odbijajući se od drugih objekata iz okoline i od čovekovog vizualnog sistema. Neki objekti, odnosno materijali reflektuju svetlost (zid, klupa, papir), dok neki propuštaju svetlost (staklo). Kod nekih objekata se javlja *apsorbovanje* dela svetlosti, transformisanjem dela svetlosti u energiju, kod nekih objekata javlja se *refleksija*, odnosno svetlosni zrak se odbija od površi pod uglom simetričnim u odnosu na normalu površi, kod nekih se pak javlja *razbacivanje* svetlosti čime se svetlost odbija od površine objekta u različitim pravcima, a kod nekih *refrakcija*, tj. *prelamanje* svetlosti, odnosno pravac kretanja svetlosti menja se usled promene brzine svetlosti (slika 9.6).

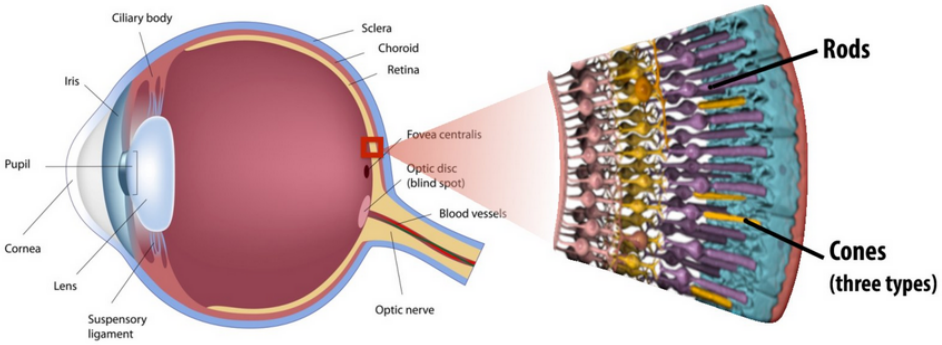


Slika 9.6: Različito ponašanje svetlosti: apsorpcija, refleksija, razbacivanje i prelamanje svetlosti.

Pojam boje veoma je složen i oslanja se na različite koncepte i rezultate na polju fizike, psihologije, umetnosti, dizajna i računarske grafike. U narednom poglavlju uvešćemo one koncepte koji su od značaja za oblast računarske grafike.

9.2 Vizualni sistem čoveka

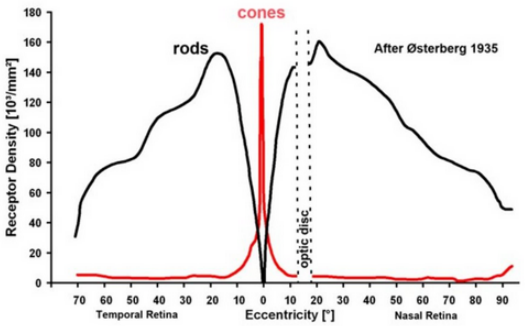
Postavlja se pitanje na koji način čovek doživljava boju koja odgovara određenom opsegu iz spektra elektromagnetnog zračenja iz opsega talasnih dužina koji odgovara vidljivoj svetlosti? Naime, zraci svetlosti ulaze kroz zenicu i stižu do neurona na zadnjem zidu oka. U zavisnosti od ugla pod kojim zraci ulaze u zenicu, oni pogađaju i stimulišu različite neurone (slika 9.7).



Slika 9.7: Anatomija oka.

Na zadnjoj strani oka nalaze se fotoreceptori. Postoje dve vrste fotoreceptora: *štapićaste ćelije* ili *štapići* (eng. rods) i *kupaste ćelije* ili *čepići* (eng. cones). Pritom prve vrste receptora ima neuporedivo više, oko 120 miliona, dok drugih ima oko 6-7 miliona. Štapići su primarni receptori u mračnim uslovima – oni su osetljivi na intenzitet svetlosti i od pomoći su u situaciji kada su svetlosni uslovi loši ili kada se nalazimo u mraku. Čepići su primarni receptori u uslovima jakog osvetljenja. Naime, u situaciji kada ima puno svetlosti, čepići su zaduženi za razlikovanje boje. Pošto receptora prve vrste ima drastično više, oko je mnogo bolje u razlikovanju intenziteta, nego u razlikovanju boja.

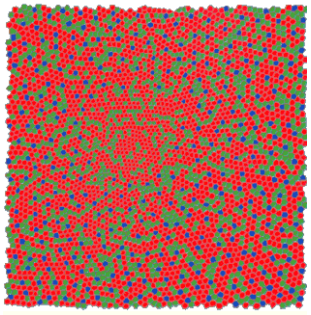
Gustina receptora u mrežnjači nije svuda ista: najveća gustina čepića je u centralnoj jami mrežnjače (lat. fovea centralis), dok kako se odaljavamo od centralne jame broj čepića je sve manji. Stoga oko najbolje razlikuje boje u središtu oblasti koju posmatramo, dok je u delu vidnog polja koje odgovara perifernom vidu razlikovanje boja jako loše. S druge strane, u centralnoj jami mrežnjače se nalazi najmanji broj štapića. Na zadnjem zidu mrežnjače nalazi se optički nerv koji povezuje oko sa mozgom: tu se nalazi slepa tačka u kojoj nema nijedne vrste receptora (slika 9.8).



Slika 9.8: Ilustracija gustine receptora u mrežnjači.

Postoje tri tipa čepića: S, M i L. Svaki od njih je osetljiv na drugačiji opseg talasnih dužina vidljive svetlosti: štapići tipa S (od engleskog short) na svetlost kratke, štapići tipa M (od engleskog middle) na svetlost srednje, a štapići tipa L (od engleskog long) na svetlost duge talasne dužine. Čepići tipa S, M i L ugrubo odgovaraju receptorima za plavu, zelenu i crvenu boju, redom. Iz tog razloga se u grafičkim aplikacijama često kompletan spektar vidljive svetlosti aproksimira sa tri vrednosti: crvenom, zelenom i plavom komponentom. Napomenimo i to da raspodela tipova čepića u oku nije ravnomerna: oko 64% su tipa L, oko 32% su tipa M, a samo 4% su tipa S (slika 9.9).

Razmotrimo uopšteno način na koji neki fotosenzor (koji može biti oko ili senzor kamere ili nešto treće) reaguje na svetlost. Svetlost na ulazu u senzor određena je raspodelom elektromagnetnog zračenja duž spektra talasnih dužina i za datu vrednost talasne dužine λ ima vrednost $\Phi(\lambda)$. *Funkcija spektralnog*

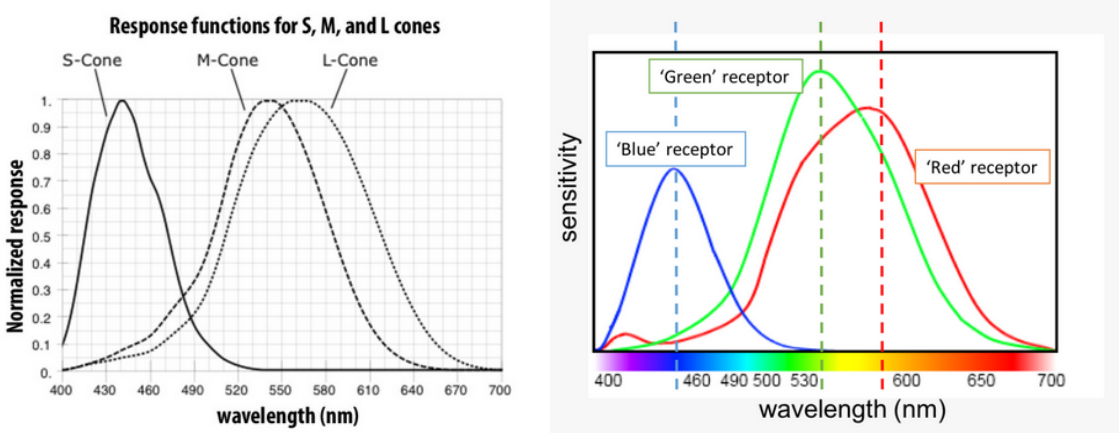


Slika 9.9: "Mozaik" čepića.

odgovora (eng. spectral response function) $f(\lambda)$ predstavlja osetljivost senzora na svetlost neke talasne dužine: ako je za neku vrednost λ vrednost $f(\lambda)$ velika, čak i mala količina svetlosti talasne duzine λ uzrokovaće jaku reakciju senzora i obratno. Odgovor fotosenzora, na primer oka, za datu talasnu dužinu svetlosti odgovara vrednosti proizvoda funkcija Φ i f za datu talasnu dužinu. Ukupan odgovor R fotosenzora biće jednak vrednosti integrala po svim talasnim dužinama proizvoda ulazne raspodele dolazne svetlosti i osetljivosti senzora na svaku od talasnih dužina:

$$R = \int_{\lambda} \Phi(\lambda) f(\lambda) d\lambda$$

U slučaju oka, funkcije spektralnog odgovora za čepiće tipa S, M i L se međusobno razlikuju. Međutim, funkcije spektralnog odgovora za M i L receptor se dosta poklapaju (slika 9.10, levo): obe se grupišu u zelenom delu spektra, pri čemu L dodaje malo crvene komponente. Na slici 9.10 (desno) je za svaku od talasnih dužina prikazano koliko receptora kog tipa reaguje na svetlost datog intenziteta.

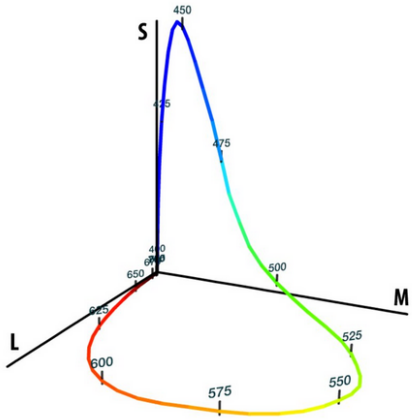


Slika 9.10: Spektralni odgovor čepića.

Na slici 9.11 ilustrovan je (S, M, L) odgovor čepića na monohromatsku svetlost određene talasne dužine: duž svake od koordinatnih osa meri se odgovor za jedan tip čepića.

Čovekovo oko ne može da meri spektar dolazeće svetlosti, niti mozak može da primi spektar kao informaciju od oka. Stoga oko meri vrednosti koje su rezultat integracije dolazećeg spektra u odnosu na funkciju odziva S, M i L čepića: $S = \int_{\lambda} \Phi(\lambda) S(\lambda) d\lambda$, $M = \int_{\lambda} \Phi(\lambda) M(\lambda) d\lambda$, $L = \int_{\lambda} \Phi(\lambda) L(\lambda) d\lambda$.

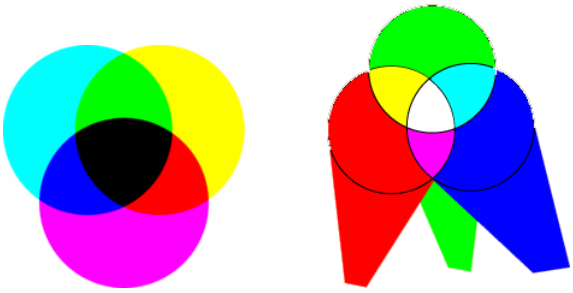
Interesantno pitanje je da li je moguće da dva različita spektra daju istu boju, tj. matematički gledano da dve različite funkcije imaju istu površinu ispod grafika. Odgovor je jasan – s obzirom da postoji mnogo više krivih nego površina, može se desiti da se dva različita spektra integrišu u isti (S,M,L) odgovor, odnosno u isti fizički odgovor u oku. Ovakve spektre nazivamo *metamerima*. Postojanje metamera može delovati kritično za reprodukciju boja, međutim, ne moramo reprodukovati baš isti spektar iz realnog života da bismo na uređaju proizveli opaženu boju.



Slika 9.11: S, M i L odgovor čepića na monohromatsku svetlost određene talasne dužine vizualizovan u vidu tačaka u 3D prostoru.

9.3 Prostor boja i kolor modeli

Već smo napomenuli da je puna informacija o boji nekog objekta kodirana njegovim spektrom, međutim, taj način kodiranja boje nije praktičan. Dakle, potrebno je osmisliti neki pogodniji i jednostavniji način kodiranja boja. Postoji veliki broj načina na koje je moguće zadati boju i oni se međusobno razlikuju po tome koji opseg boja je na taj način moguće predstaviti, koliko je izabrani način zadavanja udoban za čoveka i sl.

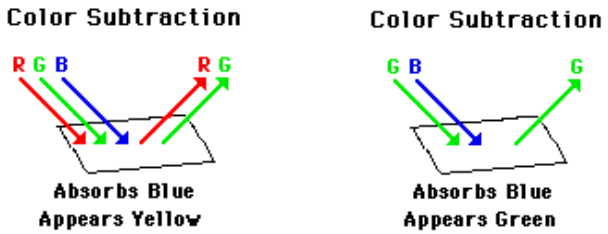


Slika 9.12: (a) Propuštanje svetlosti kroz nekoliko filtera – subtraktivno mešanje boja. (b) Usmeravanje svetlosti različitih talasnih dužina na komad belog papira – aditivno mešanje boja.

Postoji razlika u tome kako se meša svetlost, a kako pigmenti (farbe). Čovek ne vidi svetlost, već samo može da registruje svetlost koja se reflektuje od nekog objekta ili prolazi kroz njega. Svaki od objekata reflektuje određenu količinu svetlosti, ali takođe apsorbuje neki deo svetlosti. Naime, ako bismo uperili crveno i zeleno svetlo na ravnomerno reflektujući komad belog papira, reflektovana svetlost delovala bi žuto. Za razliku od toga, ako na komad belog papira nanesemo crvenu farbu, pa preko nje nanesemo zelenu, došlo bi do mešanja crvene i zelene farbe. Pritom bi se apsorbovala sva svetlost osim one koju doživljavamo kao crvenu, odnosno kao zelenu, i one bi zajedno apsorbovale skoro svu svetlost i dobijena boja bi delovala braon. Dakle, farbe se ponašaju kao filter između posmatrača i površi koja reflektuje svetlost. U skladu sa tim postoje dva koncepta mešanja boja: *aditivno* i *subtraktivno* (slika 9.12): aditivno mešanje boja odgovara mešanju svetlosti, a subtraktivno mešanju farbi. U stvari, spektar je taj koji se dodaje odnosno filtrira, dok mehanizam percepcije boja ostaje neizmenjen.

Razmotrimo primer bele svetlosti koja se dobija kao mešavina crvene, zelene i plave svetlosti. Ako se majica koja apsorbuje plavu svetlost osvetli svetlošću bele boje, samo će se crvena i zelena svetlost reflektovati od nje. Mešavinu reflektovane crvene i zelene svetlosti čovekovo oko doživljava kao žutu svetlost i čovek će zaključiti da je majica žute boje (slika 9.13 levo). Međutim, ako bismo tu istu majicu obasjali svetlošću cijan boje koja predstavlja mešavinu plave i zelene boje, oduzimanjem iz svetlosti plave boje zaključili bi da je majica zelene boje (slika 9.13 desno). Dakle, boja objekta nije u samom objektu već u svetlosti koja se reflektuje ili prelama kroz objekat. Mi košarkašku loptu vidimo kao narandžastu jer pigmenti u koži lopte odbijaju (reflektuju) svetlost žute, narandžaste i crvene boje, dok apsorbuju svetlost zelene, plave i ljubičaste boje. Aditivno mešanje boja igra važnu ulogu kod televizora

i monitora u boji i za npr. osvetljenje scene u pozorištu, dok je subtraktivno mešanje boja važno za štampu.

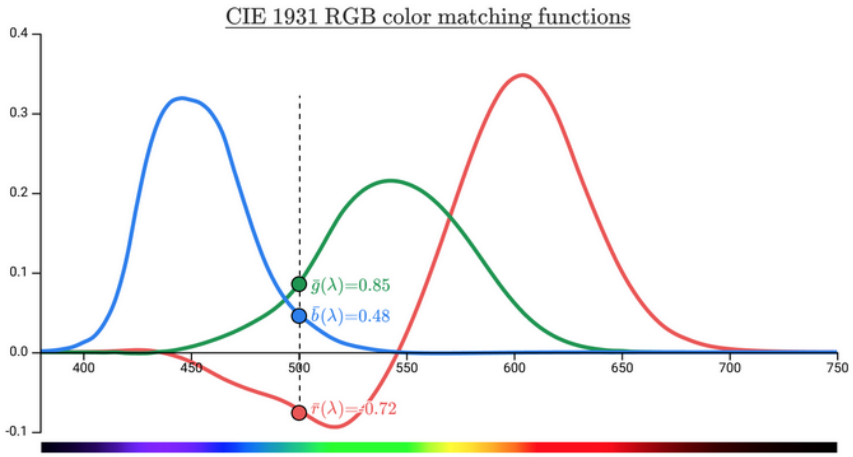


Slika 9.13: Oduzimanje boja iz spektra.

Postoji veliki broj modela za opisivanje boja. Ono što im je zajedničko jeste da su gotovo svi zasnovani na nekim trima nezavisnim karakteristikama koje se mogu biti birati na različite načine. Neki od njih su zasnovani na aditivnom, a neki na subtraktivnom mešanju boja. U nastavku ćemo razmotriti nekoliko često korišćenih modela za opis hromatske svetlosti.

9.3.1 CIE prostor boja

Može delovati da se svaka boja iz vidljivog dela spektra može predstaviti kao kombinacija crvene, zelene i plave boje (tzv. primarnih boja[1]), međutim to nije tačno. Na slici 9.14 prikazana je količina crvene, zelene i plave svetlosti potrebne da bi se dobila svaka od spektralnih boja. Primetimo da je za mnoge vrednosti talasnih dužina koeficijent uz bar jednu od boja negativan, što ukazuje na nemogućnost da se ove boje proizvedu mešanjem svetlosti crvene, zelene i plave boje. Naime, da bismo mogli da pokrijemo sve moguće vidljive vizualne senzacije bilo bi potrebno kao primarne boje uključiti sve moguće spektralne boje. To, međutim, ne znači da je pogrešna ideja da se mešanjem svetlosti crvene, zelene i plave boje proizvedu druge boje. Šta više, na ovaj način moguće je predstaviti veliki opseg boja, jer da nije tako ekrani u boji ne bi funkcionisali. Ipak, postoje i boje koje se ne mogu dobiti mešanjem svetlosti crvene, zelene i plave boje i one se ne mogu prikazati na ekranu.



Slika 9.14: Funkcije kojima se za svaku talasnu dužinu λ zadaje količina crvene, zelene i plave svetlosti koju treba pomešati kako bi se proizvela senzacija jednaka svetlosti te talasne dužine.

Razmatranjem slike 9.14 možemo zaključiti da se svetlost talasne dužine od 500nm ne može izraziti kao kombinacija svetlosti crvene, zelene i plave boje. Međutim, ako bismo svetlost talasne dužine 500nm pomešali sa svetlošću crvene boje, njihovu kombinaciju bismo mogli da izrazimo pomoću svetlosti plave i zelene boje. Sa slike 9.14 moguće je očitati koje se spektralne boje ne mogu prikazati kao kombinacija

[1]Nekad se pod primarnim bojama podrazumevaju proizvoljne tri boje za koje važi da kada se pomeša svetlost tih boja određenih intenziteta dobijemo belu boju.

crvene, zelene i plave boje. Međutim, sa njega ne vidimo koje nespektralne boje možemo, odnosno ne možemo predstaviti na ovaj način.

1931. godine Međunarodna komisija o osvetljenju (Commission Internationale de l'Éclairage, skraćeno CIE) definisala je primitive **X**, **Y** i **Z**, sa svojstvom da se sve boje vidljive čoveku mogu predstaviti kao nenegativna linearna kombinacija ove tri primitive. Kako bi ovo bilo moguće bilo je neophodno kreirati primitive koje imaju negativne regione u svom spektru, odnosno primitive koje ne odgovaraju fizički ostvarivim izvorima svetlosti. I pored toga ovako definisane primitive imaju neke prednosti:

- Ako proizvoljni izvor svetla T zapišemo kao:

$$T = c_X\mathbf{X} + c_Y\mathbf{Y} + c_Z\mathbf{Z}$$

  broj c_Y doživljavamo kao intenzitet svetla.

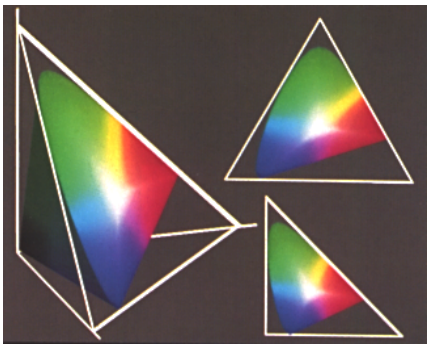
- Funkcije koje odgovaraju bojama za **X**, **Y** i **Z** su svuda nenegativne, te se sve boje mogu predstaviti kao nenegativne linearne kombinacije ovih primitiva.

- S obzirom na to da se crvena, zelena i plava primitiva mogu izraziti kao linearne kombinacije primitiva **X**, **Y** i **Z**, svaka njihova kombinacija se takođe može izraziti na ovaj način. Stoga postoji direktna veza između definicije boje korišćenjem **X**, **Y** i **Z** primitive i definicije boje u terminima crvene, zelene i plave boje (tzv. RGB modela koji će biti kasnije detaljno opisan) i obratno.

Neka su X, Y i Z težinski koeficijenti primenjeni na CIE primitive da bi se izrazila boja C, odnosno neka važi $C = X\mathbf{X} + Y\mathbf{Y} + Z\mathbf{Z}$. CIE definiše vrednosti koje zavise samo od *dominantne talasne dužine* (eng. dominant wavelength)[2] i zasićenosti, a koje su nezavisne od ukupnog intenziteta. Njih možemo dobiti deljenjem vrednosti X, Y i Z vrednošću $X + Y + Z$, koju možemo smatrati ukupnom količinom svetlosne energije:

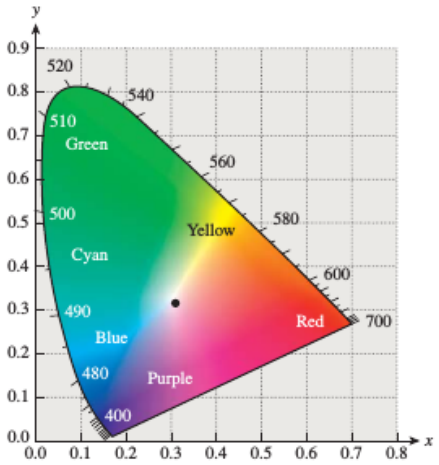
$$\begin{aligned} x &= \frac{X}{X+Y+Z} \\ y &= \frac{Y}{X+Y+Z} \\ z &= \frac{Z}{X+Y+Z} \end{aligned} \tag{9.1}$$

Naime, ako se duplira dolazno svetlo, duplira se i svaka od vrednosti X, Y i Z, ali i suma $X + Y + Z$, pa koeficijenti x, y, z ostaju nepromenjeni.



Slika 9.15: Nekoliko pogleda na ravan $x + y + z = 1$ u CIE prostoru; u donjem desnom uglu prikazana je projekcija na Oxy koordinatnu ravan.

Tri primitive x, y i z nisu nezavisne, štaviše važi $x + y + z = 1$, te ukoliko znamo vrednosti x i y možemo izračunati vrednost z (slika 9.15). Stoga se skup boja nezavisnih od intenziteta može nacrtati samo u Oxy ravni. Na taj način dobija se *CIE dijagram obojenosti* (eng. CIE chromaticity diagram), prikazan na slici 9.16. Boje koje su povezane intenzitetom se na ovom dijagramu ne prikazuju (na primer braon boja je narandžasto-crvena za malu vrednost intenziteta). CIE dijagram obojenosti ima oblik potkovice čija granica odgovara monospektralnoj svetlosti od oko 380nm do oko 700nm, odnosno odgovara vidljivoj svetlosti. Blizu centra potkovice nalazi se *izvor svetlosti* C (eng. illuminant) koji



Slika 9.16: CIE dijagram obojenosti. Granica dijagrama se sastoji od boja koje odgovaraju monospektralnoj svetlosti date talasne dužine (u nm), odnosno spektralno čistim bojama. Tačke na duži koja povezuje početak i kraj spektralnih tačaka su nespektralne: predstavljaju mešavinu dve monohromatske boje. Tačka u centru odgovara izvoru svetlosti C.

predstavlja standardnu referencu za belo na dnevnoj svetlosti. Ona je bliska tački sa koordinatama $x = y = z = 1/3$.

Primetimo da ako znamo vrednosti x i y vrednost z možemo izračunati kao $1 - x - y$, ali nam ovo ne omogućava da povratimo vrednosti X, Y i Z. Za to nam treba još neka dodatna informacija i u te svrhe se najčešće koristi vrednost Y, koja odgovara intenzitetu boje. Dakle, vrednosti X, Y i Z je na osnovnu jednačina (9.1) moguće izraziti preko vrednosti x, y i Y na sledeći način:

$$X = \frac{x}{y}Y, \quad Y = Y, \quad Z = \frac{1 - x - y}{y}Y$$

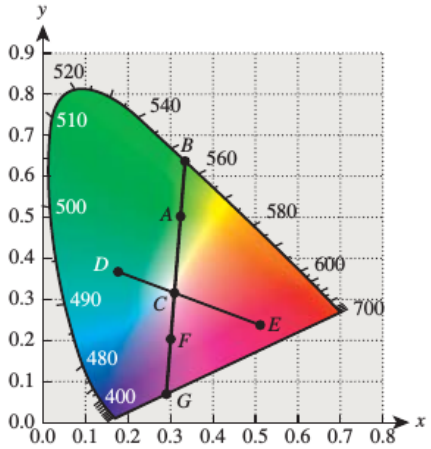
CIE dijagram obojenosti ima različite primene. Jedna od njih je definisanje dominantne talasne dužine i zasićenosti boje, odnosno njene *energetske čistoće* (eng. excitation purity). Razmotrimo boju kojoj odgovara tačka A sa slike 9.17: ona se može predstaviti kombinovanjem izvora svetlosti C sa čistom spektralnom bojom B. Dakle, tačka B definiše dominantnu talasnu dužinu boje A. Što je tačka A bliža tački B, to je ona spektralno čistija (zasićenija). Energetsku čistoću boje A, dakle, možemo definisati kao količnik dužina duži AC i BC. Definiciju možemo proširiti i na tačku C, čija se energetska čistoća postavlja na 0. Za neke boje, kao što je recimo ona koja odgovara tački F, zrak iz tačke C kroz F seče granicu dijagrama u tački G koja nije spektralna i ovakve boje nazivamo *nespektralnim* (eng. nonspectral colors) i za njih ne postoji dominantna talasna dužina. Međutim, i dalje možemo razmatrati količnik dužina duži CF i CG i na taj način i za ovakve tačke definisati energetsku čistoću.

CIE dijagram obojenosti se može koristiti i za definisanje *komplementarnih boja* (eng. complementary colors): za dve boje kažemo da su komplementarne ako se njihovim kombinovanjem može dobiti nijansa sive, poput bele i crne boje. U konteksu CIE dijagrama, potrebno je njihovim kombinovanjem dobiti izvor svetlosti C. Na primer, na slici 9.17 boje D i E bile bi komplementarne. Pritom se uobičajeno ne zahteva da mešavina bude pola-pola, jer onda za neke boje ne bi postojale komplementarne (npr. za boju B sa slike 9.17).

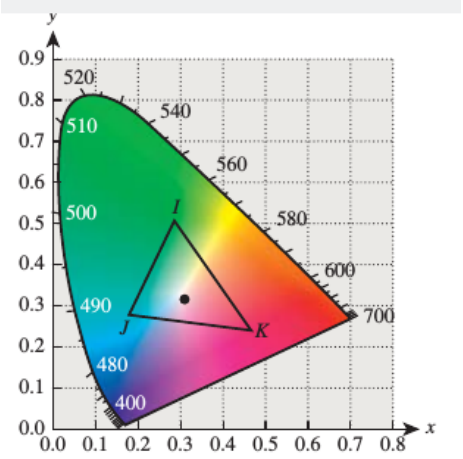
Još jedna važna primena CIE dijagrama obojenosti je za utvrđivanje *opsega boja* (eng. color gamut). Uređaj koji može da proizvede neke dve boje, može takođe da proizvede i boje koje predstavljaju konveksne (nenegativne) kombinacije te dve boje. Naime, mešanjem boja I i J sa slike 9.18 mogu se dobiti sve boje duži IJ variranjem međusobnog odnosa količine ove dve boje. Treću boju K je moguće izmešati sa različitim mešavinama boja I i J i na taj način se može dobiti proizvoljna boja iz trougla IJK, variranjem relativnih odnosa količina ove tri boje. Ako uporedimo oblik CIE dijagrama obojenosti i opsega boja koje je moguće predstaviti putem neke tri monospektralne boje, jasno je zašto se mešanjem svetlosti crvene, zelene i plave boje ne mogu dobiti sve boje vidljivog dela svetlosti (slika 9.19).

Čovekovo oko može razlikovati oko 7 miliona različitih boja. Pored opsega vidljivih boja, veoma važno pitanje je i koliko je oko osetljivo na promene boja. Za ilustrovanje ovog koncepta na CIE di-

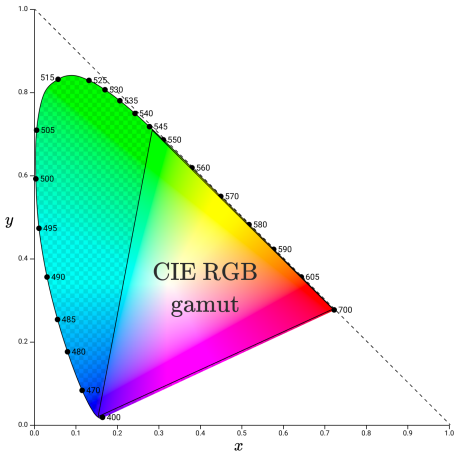
[2]Dominantna talasna dužina date nespektralne boje jeste talasna dužina spektralne boje (monohromatske svetlosti) koja odaje isti utisak nijanse boje.



Slika 9.17: Ilustracija računanja energetske čistoće i komplementarnih boja u odnosu na dijagram obojenosti.



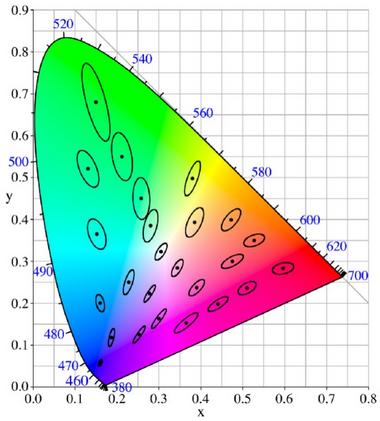
Slika 9.18: Mešanje boja na dijagramu obojenosti. Sve boje sa duži IJ mogu se napraviti mešanjem boja I i J; sve boje unutar trougla IJK mogu se napraviti mešanjem boja I, J i K.



Slika 9.19: Opseg boja koje je moguće predstaviti kao kombinaciju crvene, zelene i plave boje.

jagramu obojenosti koriste se *MacAdam elipse* koje su ilustrovane na slici 9.20. Svaka od elipsi na slici (MacAdam elipse) obuhvata region boja koje čovekovo oko ne može da razluči od boje u centru elipse. Konture elipsi predstavljaju boje čije su razlike u boji jedva primetne (eng. JND, just noticeable difference). MacAdam je izveo eksperiment za 25 različitih tačaka na CIE dijagramu i odgovarajuće elipse su

prikazane na slici. Primetimo da se veličina i orijentacija elipsi u velikoj meri razlikuje u odnosu na to u kojoj tački je izvedeno testiranje.



Slika 9.20: Ilustracija MacAdamovih elipsi (uvećanih 10 puta) koje odgovaraju bojama koje nisu razlučive od strane čovekovog oka.

9.4 Kolor modeli za rastersku grafiku

U ovom poglavlju prikazaćemo nekoliko kolor modela za rastersku grafiku koja se koriste za opis boja koje različiti uređaji mogu da proizvedu. Svi ovi modeli su ograničeni, u smislu da mogu da opišu boje do određenog intenziteta: ovo često odgovara fizičkim karakteristikama uređaja. Na primer, postoji maksimalna vrednost sjajnosti koju LCD monitor može da proizvede. Najčešće korišćeni kolor modeli su RGB, HSV i HLS koji se koriste kod monitora, YIQ model koji se koristi za TV kolor sistem i CMY kolor model koji se koristi u uređajima za štampanje. Izbor kolor modela može biti motivisan jednostavnošću (RGB model), intuitivnim zadavanjem boja (HSV i HLS model) ili određenim inženjerskim interesom (YIQ model koji se koristi za emitovanje TV signala ili CMY model za štampu).

Kolor model za rastersku grafiku zadaje se trodimenzionim koordinatnim sistemom i podoblašću oblasti svih vidljivih boja. Na primer, RGB kolor model je određen kockom u trodimenzionom prostoru. Svrha kolor modela je da omogući jednostavno adresiranje boja iz nekog opsega. Na primer, opseg boja monitora može se zadati RGB kolor modelom. Kao što smo već pomenuli, ovaj kolor model se ne može koristiti za zadavanje svih vidljivih boja.

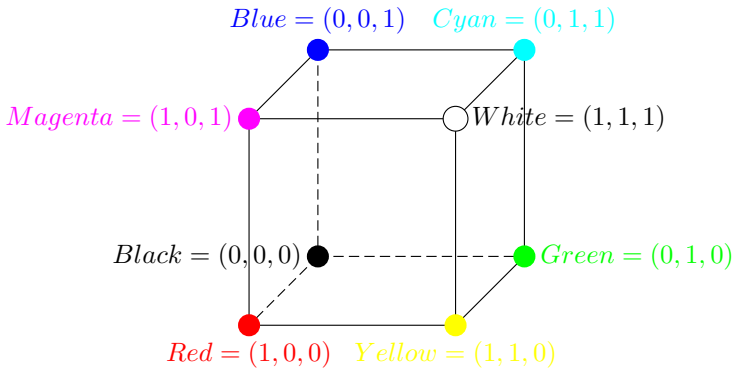
Napomenimo da u opštem slučaju konverzija između različitih kolor modela ne mora da ima smisla. Razmotrimo CMY kolor model koji je zasnovan na ideji nanošenja mastila na beli papir koji je osvetljen određenim svetlom: svetlo koje se reflektuje ne može biti sjajnije od tog osvetljenja. Stoga se ne može izvesti konvertovanje boje koja se koristi na ultrasjajnom ekranu u CMY kolor model. Prema tome, prilikom kreiranja slike u računarskoj grafici, trebalo bi omogućiti čuvanje slike bez gubitaka sa svim bitnim informacijama, kako bi je bilo moguće naknadno konvertovati u neki drugi format.

9.4.1 RGB kolor model

U RGB kolor modelu boje se predstavljaju kao kombinacija crvene, zelene i plave boje, pri čemu vrednosti uz svaku od boja pripadaju opsegu $[0, 1]$. RGB kolor model se koristi za ekrane u boji. Primitive koje se koriste u ovom modelu su *aditivne* – pojedinačni doprinosi svake primitive se dodaju i daju konačni rezultat.

RGB kolor model se zadaje jediničnom kockom u trodimenzionom prostoru tako da se u temenu kocke sa koordinatama $(1, 0, 0)$ nalazi crvena boja, u temenu sa koordinatama $(0, 1, 0)$ zelena, a u $(0, 0, 1)$ plava boja. U koordinatnom početku nalazi se crna boja, dok se bela boja nalazi u temenu kocke sa koordinatama $(1, 1, 1)$. Komplementarne boje nalaze se jedna nasuprot druge. Nijanse sive se nalaze duž glavne dijagonale kocke, dok se sa većim udaljavanjem od glavne dijagonale dobijaju sve zasićenije boje (slika 9.21).

Dva ekrana mogu imati različite opsege boja, pa se vrednosti RGB primitiva moraju interpretirati u zavisnosti od konkretnog uređaja. Prelazak sa RGB komponenti opsega boja jednog ekrana na vrednosti RGB komponenti opsega boja drugog ekrana može se zadati matricom dimenzije 3×3. Nekad se može dogoditi da se neka RGB trojka za jedan ekran nakon množenja matricom transformacije transformiše



Slika 9.21: RGB kolor model.

u trojku čije su neke vrednosti veće od 1 ili manje od 0 – ovo ukazuje na to da postoji boja u opsegu boja prvog ekrana koja je van opsega boja drugog ekrana. Ovaj problem se može rešiti ili ignorisanjem matrice transformacije ili zaokruživanjem vrednosti tako da se vrednosti manje od 0 tumače kao 0, a vrednosti veće od 1 kao 1, ili nekim naprednijim metodama: npr. kompesovanjem opsega boja prvog ekrana skaliranjem u odnosu na centar opsega tog ekrana, tako da se sve boje koje treba prikazati na prvom ekranu preslikavaju u boje koje je moguće prikazati na drugom ekranu.

9.4.2 CMY i CMYK kolor model

Cijan (zeleno-plava), magenta (ljubičasto-crvena) i žuta boja predstavljaju komplementarne boje crvenoj, zelenoj i plavoj boji, redom. Kada se koriste kao filteri da oduzmu boju iz svetlosti bele boje, nazivaju se *subtraktivnim primitivama*.

U CMY kolor modelu boje se opisuju kao mešavine cijan, magenta i žute boje. Ovaj kolor model se koristi za štampače, jer mastilo reflektuje neki deo primljene svetlosti, a apsorbuje drugi deo. Kada se površ prekrije cijan mastilom, iz reflektovane bele svetlosti (koja predstavlja sumu crvene, zelene i plave svetlosti) apsorbuje se crveno svetlo, a reflektuje plavo i zeleno, magenta apsorbuje zeleno, a žuta plavo svetlo. Kada se pomešaju dva mastila, svetlost koja se reflektuje je ona koju ne apsorbuje nijedna od njih. Stoga, mešavina mastila cijan i magente apsorbuje i crvenu i zelenu boju, dajući pritom refleksiju plave svetlosti. Ako izmešamo cijan, magenta i žuto mastilo, apsorbovaće se crvena, zelena i plava svetlost i dobićemo crnu boju. U stvarnosti ta boja nije sasvim crna, jer mešavina cijan, magente i žute boje ne uspeva da apsorbuje baš svu svetlost, stoga štampači često imaju i četvrto mastilo – crno (u oznaci K) koje se koristi da zameni tamnije mešavine cijan, magenta i žute boje.

CMY kolor model je, poput RGB kolor modela, određen jediničnom kockom u trodimenzionom prostoru, tako da se u koordinatnom početku umesto crne boje nalazi bela boja, cijan se nalazi u temenu kocke sa koordinatama $(1, 0, 0)$, magenta u temenu sa koordinatama $(0, 1, 0)$, a žuta boja u temenu sa koordinatama $(0, 0, 1)$ (slika 9.22).

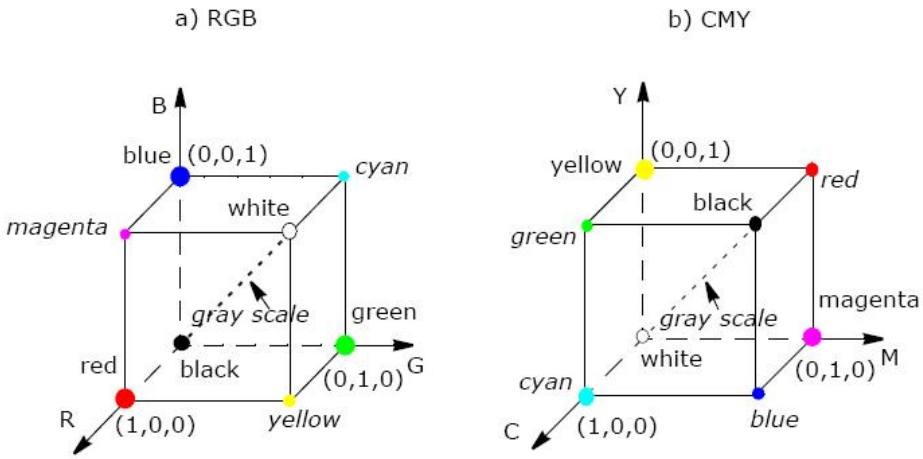
Prelazak sa RGB komponenti na CMY komponente može se opisati narednom jednačinom:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Prelazak iz vrednosti u CMY modelu na vrednosti u RGB modelu se opisuje analognom jednačinom. Jedinični vektor kolona u RGB modelu odgovara beloj boji, a u CMY modelu crnoj boji.

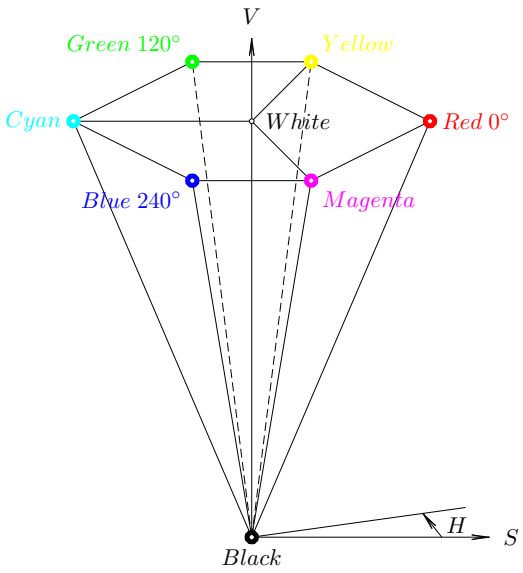
9.4.3 HSV i HLS kolor modeli

Zadavanje boja putem RGB kolor modela nije intuitivno za korisnika jer individualne komponente ovog modela ne odgovaraju čovekovoj percepciji boje, odnosno ne oslikavaju njenu nijansu, niti koliko je boja svetla, a koliko zasićena i slično. Na primer, prilikom izmene boje od crvene ka narandžastoj malim povećanjem zelene komponente boja postaje i svetlija, a ono što bismo mi na taj način želeli da postignemo jeste da samo izmenimo nijansu boje. Slično, čoveku visoko zasićene boje izgledaju čisto,



Slika 9.22: Uporedni prikaz RGB i CMY kolor modela (preuzeto sa https://scc.ustc.edu.cn/zlsc/sugon/intel/ipp/ipp_manual/IPPI/ippi_ch6/ch6_color_models.htm).

dok manje zasićene boje deluju "isprano" i bilo bi zgodno omogućiti da se u modelu eksplicitno zadaje i ova karakteristika boje.



Slika 9.23: HSV kolor model.

Postoje dva kolor modela kojima se boja zadaje u terminima ovih karakteristika i to su HSV i HLS kolor model. HSV kolor model koristi cilindrični koordinatni sistem i podskup skupa boja koje je moguće na ovaj način adresirati zadaje se šestostranom piramidom[3] (slika 9.23). Komponente HSV kolor modela su:

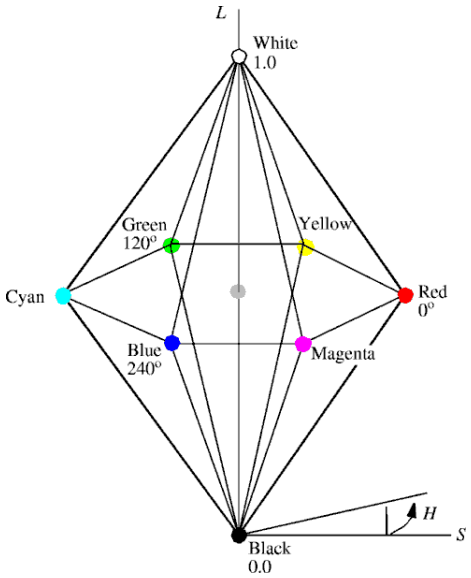
- H: *nijansa* (eng. hue) – karakteriše boju: da li je u pitanju crvenkasta, zelenkasta, žućkasta boja, ... ($0° - 360°$),
- S: *zasićenost* (eng. saturation) – karakteriše koliko se boja razlikuje od sive istog intenziteta ($0\% - 100\%$ ili $0 - 1$),
- V: *sjajnost* (eng. value, brightness) – karakteriše koliko je boja svetla, odnosno tamna ($0\% - 100\%$ ili $0 - 1$).

Osnova piramide odgovara vrednosti $V = 1$ i ona sadrži relativno svetle boje. Nijansa boje se meri uglom oko vertikalne ose piramide, pri čemu crvena boja odgovara vrednosti ugla od $0°$, zelena vred-

[3]HSV kolor model se nekada opisuje i kupom umesto šestostrane piramide.

nosti $120°$, a plava uglu od $240°$. Komplementarne boje se nalaze jedna nasuprot drugoj (udaljene su za $180°$). Zasićenost boje meri se rastojanjem od vertikalne ose (visine) piramide: tačke sa visine imaju zasićenost 0, dok tačke na stranicama šestostrane piramide imaju zasićenost 1. Sjajnost boje se meri udaljenošću tačke od vrha piramide: vrh piramide odgovara crnoj boji za koju važi da je $V = 0$, dok vrednosti komponenti H i S nisu relevantne. Tačka za koju važi $S = 0$ i $V = 1$ odgovara beloj boji. Međuvrednosti komponente V za $S = 0$ odgovaraju nijansama sive. Kada je zasićenost jednaka 0, vrednost nijanse nije od značaja.

Za razliku od HSV modela, HLS kolor model pored nijanse i zasićenosti, kao treću komponentu, umesto sjajnosti, ima informaciju koliko je boja svetla, odnosno njenu *osvetljenost* (eng. lightness). I ovaj kolor model se zadaje cilindričnim koordinatnim sistemom, a opseg boja koje se mogu adresirati odgovara dvostrukoj šestrostranoj piramidi[4] (slika 9.24). Bela boja je maksimalno svetla ($L = 1$), a crna maksimalno tamna ($L = 0$). Za nijanse sive važi $S = 0$, a maksimalno zasićene boje odgovaraju vrednostima $S = 1$ i $L = 0.5$. Komponenta H koja se odnosi na nijansu boje se u ova dva modela odnosi na isti atribut, dok se pojam zasićenosti u ova dva modela razlikuje: u HLS modelu se izmenom S komponente ide od potpuno zasićene boje do ekvivalentne sive, dok se u HSV modelu izmenom S komponente ide od maksimalno zasićene boje do bele, što može delovati donekle neintuitivno.



Slika 9.24: HLS kolor model.

Postoje jednostavne formule za preračunavanje HSV i HLS komponenti u RGB komponente i obratno, ali se njima u ovom materijalu nećemo baviti.

9.5 Prikazivanje boja piksela na ekranu

9.5.1 Čovekovo opažanje intenziteta svetlosti

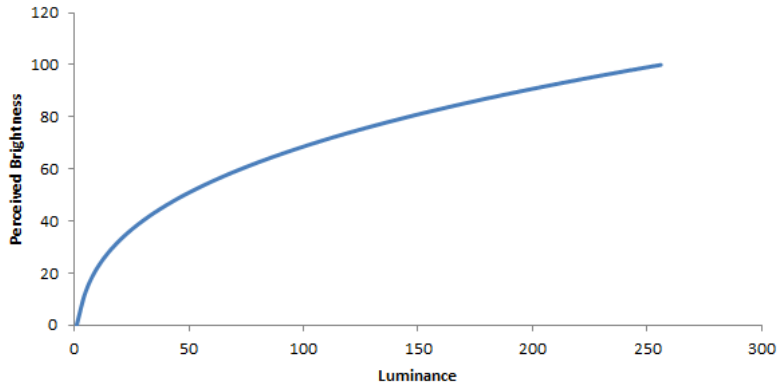
Kvantitet svetlosti može biti razmatran kao fizički pojam energije, kada se govori o *intenzitetu* ili *luminanciji* (eng. luminance), ili kao psihološki fenomen, kada se govori o *sjajnosti* (eng. brightness). Vrednost intenziteta svetlosti opisivaćemo jednim skalarom, u opsegu od 0 do 1.

Pretpostavimo da na skali od 0 do 1 želimo da predstavimo 256 različitih vrednosti intenziteta, tako da raspodela vrednosti bude u nekom smislu ravnomerna. Postavlja se pitanje kojih 256 vrednosti treba izabrati: sigurni smo da nije dobro da izaberemo 128 vrednosti u opsegu $[0, 0.1]$ i još 128 u opsegu $[0.9, 1]$ jer bi prelaz sa vrednosti 0.1 na vrednost 0.9 bio primetan.

Treba voditi računa o činjenici da je čovekovo oko osetljivije na odnose intenziteta svetlosti, nego na razlike u vrednosti intenziteta svetlosti. Naime, čovekovom oku se čini da se intenziteti 0.10 i 0.11 razlikuju jednako kao i intenziteti 0.50 i 0.55. Dakle, intenzitet svetlosti treba da bude opisan logaritamskom (a ne linearnom) skalom kako bi se dobili jednaki koraci u sjajnosti (slika 9.25).

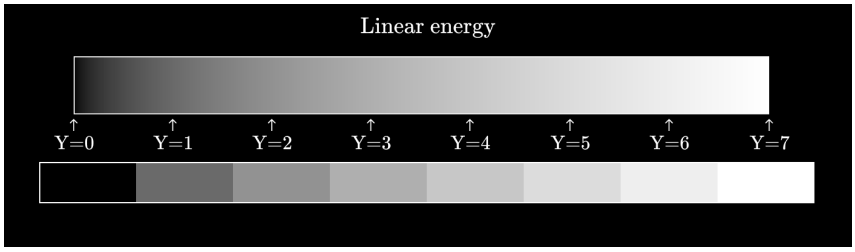
Ilustrujmo ovaj fenomen. Razmotrimo vrednosti sive koje su dobijene izborom linearne funkcije intenziteta prikazane na slici 9.26: može se primetiti da čovek razliku između vrednosti $Y = 0$ i $Y = 1$

[4] HLS model se nekada zadaje i dvostrukom kupom umesto šestostrane piramide.

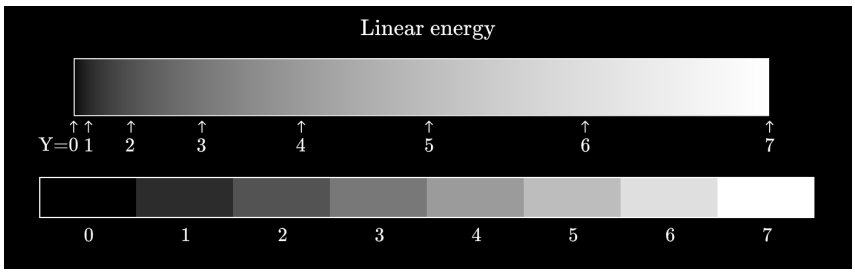


Slika 9.25: Sjajnost je logaritamska funkcija intenziteta svetlosti.

doživljava kao značajno veću nego razliku između vrednosti $Y = 6$ i $Y = 7$. Ako bismo umesto toga razmatrali stepenu funkciju, imali bismo osećaj da je raspodela nijansi sive bliska uniformnoj (slika 9.27).



Slika 9.26: Nijanse sive dobijene kao linearna funkcija energije (preuzeto sa http://jamie-wong.com/post/color/).



Slika 9.27: Nijanse sive dobijene kao stepena funkcija energije (preuzeto sa http://jamie-wong.com/post/color/).

Na koji način bi se odredilo 256 različitih ravnomerno raspodeljenih intenziteta počev od najniže vrednosti intenziteta I_0 pa do maksimalnog intenziteta 1.0? Potrebno je da postoji vrednost r koja odražava odnos dva susedna intenziteta na sledeći način:

$$I_0 = I_0,\ I_1 = rI_0,\ I_2 = rI_1 = r^2 I_0, \ldots,\ I_{255} = r^{255} I_0 = 1$$

$$r = (1/I_0)^{1/255},\ I_j = r^j I_0 = I_0^{(255-j)/255}$$

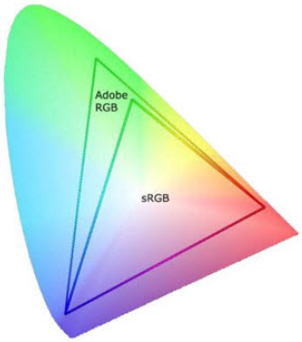
U opštem slučaju, za $n+1$ intenziteta svetlosti koji bi čoveku delovali ravnomerno raspodeljeni treba da važi:

$$r = (1/I_0)^{1/n},\ I_j = r^j I_0 = I_0^{(n-j)/n}$$

Nameće se pitanje: koliko različitih vrednosti intenziteta je dovoljno da bi se reprodukovala crno-bela slika tako da prelaz između nijansi deluje neprekidno. Pokazuje se da čovekovo oko primećuje prelaz sa tamnije na svetliju nijansu sive ukoliko je odnos intenziteta veći od 1.01. Na osnovu te vrednosti može se odrediti minimum intenziteta koji mogu da daju prihvatljivu crno-belu sliku. Za većinu uređaja, ta (idealizovana) vrednost iznosi $n = 64$.

9.5.2 Gama korekcija

Na boje u računaru se referiše korišćenjem RGB kolor modela, tj. putem vrednosti crvene, zelene i plave komponente boje. *sRGB prostor boja* (eng. standard RGB) je standardni RGB prostor boja, zajednički predložen od strane kompanija HP i Majkrosoft, koji je u širokoj upotrebi u oblasti digitalnih slika. sRGB prostor boja je dizajniran sa ciljem da obezbedi konzistentnu reprezentaciju boja na različitim uređajima i platformama. sRGB prostor boja ima značajno ograničen opseg boja koje je moguće predstaviti u poređenju sa kompletnim RGB prostorom i u njemu nedostaju visoko zasićene boje. Drugi značajan RGB prostor boja je Adobe RGB prostor boja i on značajno nadmašuje sRGB u pogledu opsega boja koje je moguće predstaviti (slika 9.28).



Slika 9.28: Odnos sRGB i Adobe RGB prostora boja.

Razmotrimo detaljnije način na koji se izračunate boje piksela, u terminima količine crvene, zelene i plave boje iz sRGB prostora boja, prikazuju na ekranu.

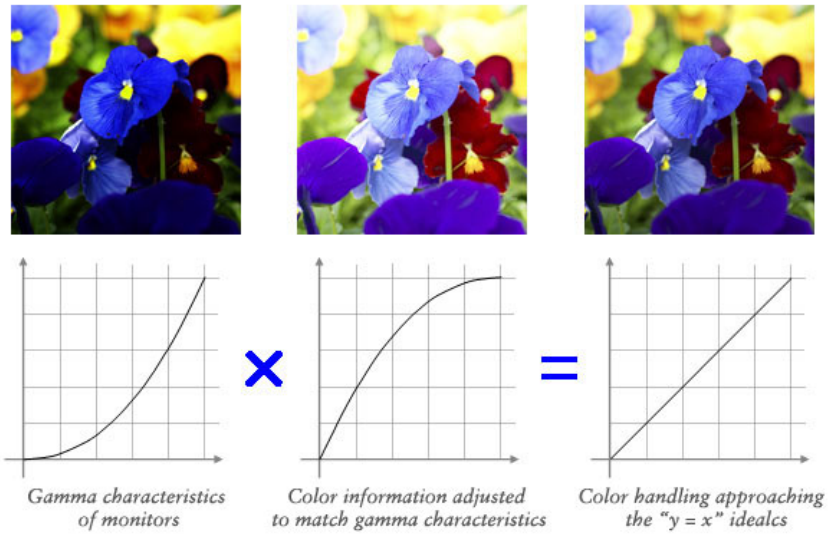
U ranija vremena ekrani su bili sa katodnim cevima, takozvani CRT ekrani. Prikazivanje odgovarajućih intenziteta boja na CRT ekranima nije trivijalna stvar zbog toga što je transformacija napona elektronskog zraka u intenzitet svetlosti na ekranu nelinearna. To znači da ako se neka komponenta boje, npr. crvena komponenta dvostruko poveća, ona ne proizvodi dvostruko veći intenzitet emitovane crvene svetlosti. Naime, intenzitet koji CRT ekran emituje je proporcionalan sa V^γ, pri čemu je sa V označen napon, dok se vrednost γ naziva *gama* i njena približna vrednost kod CRT ekrana iznosi 2.5. Dakle, kod CRT ekrana i ranih verzija LCD ekrana gama je bila direktno povezana sa naponom i igrala je važnu ulogu u preciznom reprodukovanju slika na ekranima.

Gama je važna karakteristika gotovo svih digitalnih sistema koji rade sa slikama. Na modernim LCD ekranima se parametrom gama reguliše u kojoj meri se povećanje vrednosti intenziteta boje u memoriji oslikava na povećanje osvetljenosti na ekranu. Gama definiše odnos između numeričke vrednosti piksela i njegove stvarne osvetljenosti. Naime, naše oči ne opažaju svetlost na isti način kao što to radi digitalni fotoaparat. Kod fotoaparata, kada dvostruko veći broj fotona pogodi senzor, on prima dvostruko veći signal (važi linearni odnos). Međutim, vizuelni sistem čoveka ne funkcioniše na isti način: oko dvostruko veću svetlost opaža kao neznatno svetliju (odnos nije linearan). Oko nesrazmerno opaža promene vrednosti RGB komponenti. Drugim rečima, razlika u opaženoj osvetljenosti između dve boje u RGB prostoru boja nije nužno proporcionalna razlici numeričkih vrednosti R, G i B komponente. Na primer, dve boje koje se u RGB prostoru nalaze jako blizu, prilikom prikaza jedne do druge mogu delovati veoma različito osvetljene. Dakle, osvetljenost koju čovekovo oko opaža je nelinearna funkcija količine svetlosti. Dodatno, naš vizuelni sistem je u normalnim uslovima osvetljenja mnogo osetljiviji na promene tamnih tonova nego na slične promene svetlih tonova.

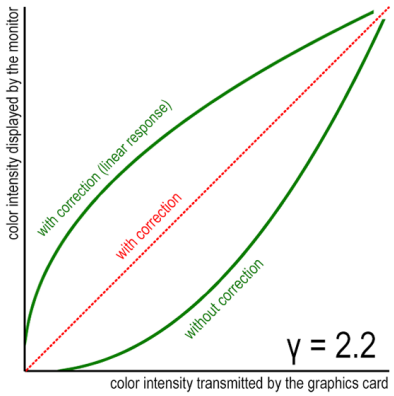
Percepcija osvetljenosti kod čoveka prati približno stepenu funkciju (slika 9.25). Primetimo da ako promenimo vrednost ulaza sa 0 na 10, izlaz se menja od 0 do oko 20, dok izmena vrednosti sa 240 na 255 zapravo ne menja vrednost na izlazu. Ali digitalni fotoapart ne funkcioniše po istom principu. Za razliku od čovekove percepcije, on prati linearni odnos. Stoga, prilikom prikazivanja slike može doći do problema. Naime, za proizvoljni ulazni signal, recimo sa fotoaparata, izlaz će biti transformisan za γ zbog nelinearnog odnosa intenziteta i napona na ekranu. Ovo rezultira slikama koje su tamnije od predviđenih. Kako bismo ovo ispravili, potrebno je na neki način korigovati ulazni signal (slika 9.29). Za posredovanje između osetljivosti našeg oka na svetlost i osetljivosti fotoaparata koristi se gama.

Gama korekcija (eng. gamma correction) je nelinearna operacija koja se koristi za kodiranje i dekodiranje vrednosti osvetljenosti na slikama i video zapisima. Iako je gama korekcija prvobitno razvijena za potrebe prevazilaženja razlika u ulaznom i izlaznom signalu CRT ekrana i za veran prikaz osvetljenosti na CRT ekranima, u današnje moderno doba to nije njena prevashodna namena. Gama korekcija se

danas koristi kako bi se slike prilagodile karakteristikama čovekovog vizualnog sistema i proizvela senzacija odgovarajućih boja. Ako se slika prikaže na računaru bez gama korekcije, korisnik će je doživeti kao ispranu ili previše svetlu. Gama se koristi da obezbedi da odnosi vrednosti na ulazu odgovaraju izlazu računara, stvarajući pravu boju. Ideja gama korekcije je da se primeni inverzna gama ekrana na konačnu izlaznu boju (time boje postaju svetlije) pre nego što se slika prikaže na ekranu. Ideja je posvetliti boje koje će se kada ih montor potamni vratiti na željene vrednosti (slika 9.29). Dakle, gama korekcija predstavlja poništavanje nepogodnog rada ekrana, odnosno korekciju informacija o bojama. Na ovaj način gama kodiranu sliku transformišemo u svetlost iz originalne scene. Gama korekcija suštinski vrši stepenovanje vrednosti intenziteta boje na eksponent $1/\gamma$, tako da kada ekran podigne vrednosti na eksponent γ, efekat ove dve operacije se poništava, a rezultat je boja koju smo i želeli da prikažemo. Kada se digitalna slika sačuva, ona je stoga "gama kodirana" - tako da dvostruko veća vrednost u datoteci odgovara onome što bismo mi smatrali dvostruko svetlijim. Većina datoteka koje čuvaju slike kodira slike sa vrednošću game od $\frac{1}{2.2} \approx 0.45$, dok se slike dekodiraju korišćenjem recipročne vrednosti game od 2.2.



Slika 9.29: Kombinovanje gama karakteristike monitora i inverzne transformacije na boju kako bi se opazile željene vrednosti o boji (preuzeto sa https://www.eizo.com/library/basics/lcd_display_gamma/).



Slika 9.30: Preslikavanje intenziteta boja koje se šalju ekranu koje vrši grafička kartica i intenziteta koje ekran prikazuje (preuzeto sa https://gamedevelopment.tutsplus.com/gamma-correction-and-why-it-matters--gamedev-14466a).

Na slici 9.30 ilustrovani su različiti scenariji prilikom korišćenja game: donja zelena kriva ilustruje situaciju kada se gama ne uzima u obzir – tada je kriva eksponencijalna. Ako izvršimo gama korekciju, stvarni odgovor će biti linearan, kakav i treba da bude. Poređenja radi, na istoj slici je prikazano i kako

bi izgledao grafik ako bismo sproveli gama korekciju, ali ekran ima linearan odziv. U ovoj situaciji će intenziteti biti deformisani na suprotan način i možemo videti da kada ih nelinenaran ekran izmeni, to se poništava i na kraju dobijamo pravu $y = x$.

Kao što smo već pomenuli na boje se u računaru referiše kao na trojku RGB vrednosti, koje redom odgovaraju vrednosti crvene, zelene i plave komponente u sRGB prostoru boja. Važno pitanje jeste na koji način na osnovu ove tri vrednosti odrediti osvetljenost boje. Opažena osvetljenost boje može se izračunati na sledeći način: najpre se sRGB vrednosti konvertuju u vrednosti iz intervala $[0, 1]$:

$$\begin{aligned} R &= R'/255 \\ G &= G'/255 \\ B &= B'/255 \end{aligned}$$

Ekran tumači svoje ulazne podatke kao da su kodirane u sRGB prostoru boja. Prilikom prikaza podataka na ekranu potrebno je linearizovati dobijene vrednosti crvene, zelene i plave komponente. Najjednostavniji način da se ovo uradi jeste da se na svaki kanal za boju primeni stepenovanje na eksponent 2.2:

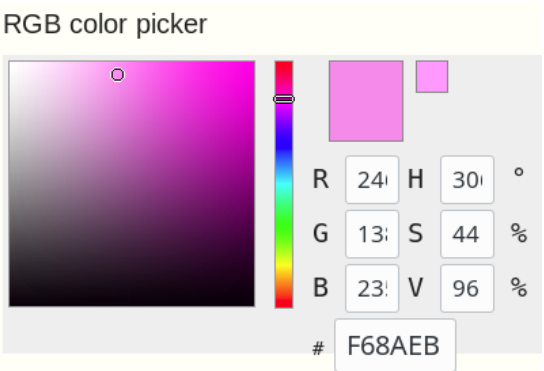
$$\begin{aligned} R_{lin} &= R^{2.2} \\ G_{lin} &= G^{2.2} \\ B_{lin} &= B^{2.2} \end{aligned}$$

Pokazuje se da najveći uticaj (čak 71%) na osvetljenost koju čovek opaža ima komponenta svetlosti zelene boje, dok svetlost plave boje ima najmanji doprinos. Najzad, osvetljenost Y dobijamo na osnovu naredne formule:

$$Y = 0.2126 \cdot R_{lin} + 0.7152 \cdot G_{lin} + 0.0722 \cdot B_{lin}$$

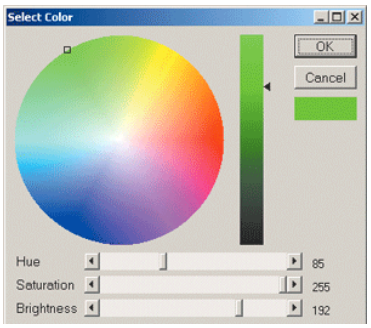
Dakle, vrednosti osvetljenosti fragmenata objekata u memoriji direktno odgovaraju broju fotona koji pogađa senzor fotoaparata kojim se slika scena koju posmatramo. Ako se na objekat koji obasjava jedan izvor svetlosti uperi još jedan identičan izvor svetla, u memoriji će se vrednost osvetljenosti tih fragmenata dvostruko povećati. Pre nego što se prikaže odgovarajuća boja na ekranu, vrši se gama korekcija kako bi slika delovala verodostojnije.

9.6 Izbor i korišćenje boja



Slika 9.31: Standardni RGB dijalog u kojem je moguće odabrati željenu boju i za nju se prikazuju vrednosti komponenti u RGB i HSV sistemu.

U kontekstu računarske grafike boje su značajne iz estetskih razloga, koriste se za postizanje realizma na slikama, za naglašavanje, a često se njima postiže i bolja funkcionalnost i i interfejs različitih aplikacija. Mnogi programi dozvoljavaju korisniku da zada boju elemenata kao što su slova, prave, površi i slično. Ne postoji jedinstveni sistem zadavanja boja koji je najzgodniji svim korisnicima. Neki programi omogućavaju korisniku da izabere boju korišćenjem različitih modova: direktnim zadavanjem RGB komponenti, zadavanjem komponenti korišćenjem slajdera, klikom na odabir boje iz prikaza spektra boja u vidu diska i sl. Korisniku se može omogućiti i da interaktivno bira boju tako što direktno intreraguje sa reprezentacijom prostora boja (slike 9.31 i 9.32).



Slika 9.32: HSV dijalog za odabir boje: prikaz boja za određenu vrednost sjajnosti, nijansa odgovara veličini ugla, a zasićenost poluprečniku.

Prilikom odabira skupa boja koje će biti korišćene u izradi neke korisničke aplikacije ili neke računarske animacije, treba voditi računa da kombinacija više boja bude što skladnija. Naime, loša kombinacija boja može da učini prikaz manje ilustrativnim od odgovarajućeg monohromatskog prikaza. Takođe, različitim odabirom kombinacija boja moguće je postići različite optičke efekte (slika 9.33).

Slika 9.33: Dva siva kvadrata na vrhu deluju da su različite boje; siva traka na dnu je iste boje čitavom dužinom.

Interpoliracija boja je važna u problemima kao što su Guroovo senčenje ili antialiasing tehnike. Boja dobijena kao rezultat interpolacije zavisiće od kolor modela u kom se interpolacija vrši. Naime, linearnom prelazu između dve tačke u jednom modelu ne odgovara nužno linearni prelazak između dve tačke u drugom modelu. Na primer, ako se u RGB kolor modelu i u HSV kolor modelu vrši interpolacija crvene i zelene boje sa težinama 0.5 dobiće se različit rezultat (bilo da se prvo radi konverzija pa interpolacija ili obratno). U slučaju Guroovog senčenja prihvatljivo je raditi interpolaciju u proizvoljnom modelu jer su boje koje se interpoliraju bliske. Međutim, u slučaju kada su boje udaljene postoji veći broj mogućnosti i nijedan od njih nije univerzalno najbolji.

Primer 9.1. Razmotrimo problem reprodukovanja boja na izlaznom uređaju koji koristi manji broj bitova za boju po pikselu. Slika u boji je zadata u terminima n bitova po pikselu, a treba je što vernije prikazati na uređaju koji podržava $m < n$ bitova po pikselu. Postavlja se pitanje kojih 2^m boja izabrati i na koji način zadati preslikavanje iz skupa od 2^n boja u skup od 2^m boja. Jednostavno rešenje je koristiti predefinisani skup prikaznih boja i fiksno preslikavanje boja sa slike u boje prikaznog uređaja. Na primer, u slučaju kada je $m = 8$, obično se koristi po 3 bita za crvenu i zelenu komponentu, a 2 za plavu (jer je oko manje osetljivo na plavu boju). Postoje i brojni drugi napredniji algoritmi koji se prilagođavaju raspodeli vrednosti komponenti polazne slike.

9.7 Pitanja

9.1 Koju boju nazivamo monohromatskom?

9.2 Da li je čovekovo oko jednako osetljivo na sve talasne dužine?

9.3 Koja svojstva ima ahromatska svetlost?

9.4 Kakvom skalom je opisan intenzitet svetlosti i zašto?

9.5 Ako je potrebno zadati n intenziteta svetlosti počev od I_0 do 1, po kojoj formuli se računa intenzitet I_j u funkciji početne vrednosti intenziteta I_0?

9.6 Šta je gama korekcija?

9.7 Opisati aditivno i subtraktivno mešanje boja.

9.8 Da li je sve vidljive boje moguće predstaviti kao nenegativnu linearnu kombinaciju
(a) crvene, zelene i plave svetlosti (da) (ne)
(b) X, Y i Z primitive (da) (ne)
(c) bilo koje tri monohromatske svetlosti (da) (ne)?

9.9 Čemu odgovara primitiva Y u CIE modelu?

9.10 Kako se definišu vrednosti x, y i z na osnovu vrednosti X, Y i Z primitiva?

9.11 Da li su vrednosti x, y i z kod CIE modela međusobno nezavisne?

9.12 Definisati CIE dijagram obojenosti. Da li se na njemu prikazuju sve boje?

9.13 Da li je granica CIE dijagrama obojenosti sačinjena samo od monohromatskih boja?

9.14 Šta je kolor model? U koje svrhe se koristi?

9.15 Zašto postoji veliki broj kolor modela?

9.16 Nabrojati kolor modele i za svaki od njih nabrojati kojim se komponentama zadaje.

9.17 Da li je CMY kolor model zasnovan na aditivnom ili subtraktivnom mešanju boja?

9.18 Gde se nalaze nijanse sive u (a) RGB modelu (b) CMY modelu (c) HSV modelu?

9.19 Gde se nalaze komplementarne boje u (a) RGB modelu (b) CMY modelu (c) HSV modelu?

Glava 10

Prostorne strukture podataka

Osnovna ideja nerekurzivnog rej kasting algoritma se zasniva na tome da se za svaki piksel generiše po jedan zrak iz otvora kamere i da se odredi prvi presek tog zraka i neke primitive sa scene. Ukoliko scena sadrži N primitiva, složenost direktnog algoritma za određivanje prvog preseka sa datim zrakom je $O(N)$. Broj zraka koje razmatramo odgovara broju piksela slike i može biti veoma veliki. S druge strane, scene mogu jako složene, sa jako velikim brojem primiiva. Zbog svega ovoga, linearna složenost po zraku nije praktično zadovoljavajuća. Postavlja se pitanje da li je na neki način moguće realizovati upit preseka u vremenskoj složenosti koja je manja od linearne. Pokazuje se da je moguće popraviti prosečnu složenost upita preseka, pogodnim strukturiranjem primitiva na sceni.

Prostorne strukture podataka (eng. spatial data structures) predstavljaju klasu struktura podataka kojima se prostorni podaci organizaciju na osnovu svog položaja i zapremine scene koju zauzimaju. One predstavljaju višedimenziono uopštenje klasičnih uređenih struktura podataka, kao što su uređena binarna stabla (eng. binary search tree). S obzirom na to da prostorne strukture podataka imaju za cilj povećanje prostora za skladištenje podataka, a da se pritom smanji vreme izvršavanja upita, poznate su i pod nazivom prostorne ubrzavajuće strukture podataka (eng. spatial acceleration data structures). One mogu ubrzati operacije kao što su pronalaženje preseka između različitih geometrijskih figura, detekcija kolizije, renderovanje u realnom vremenu i slično. Prostorne strukture podataka se, iako primarno razvijene za potrebe renderovanja i animacije u računarskoj grafici, danas koriste i u drugim oblastima – mašinskom učenju, statističkim algoritmima i dr.

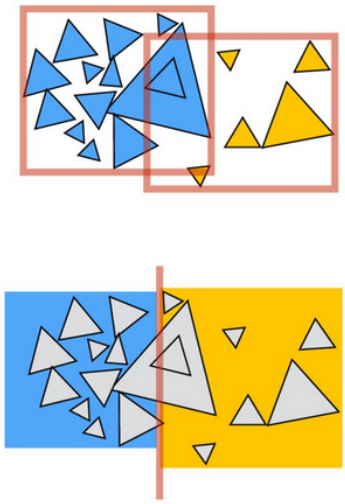
Mnogi algoritmi u računarskoj grafici se oslanjaju na upite koji se mogu opisati geometrijskim presecima. Poželjno je da vremenska složenost algoritma za određivanje preseka bude manja od linearne u funkciji broja primitiva na sceni. Pokazuje se da se vremenska složenost upita preseka na sceni koja sadrži N primitiva korišćenje prostornih struktura podataka može smanjiti na $O(\log N)$ u prosečnom slučaju.

Postoje dva različita pristupa strukturiranju primitiva na sceni koja vode različitim prostornim strukturama podataka. U prvom pristupu vrši se particionisanje primitiva u disjunktne podskupove. Jedan primer ovakve strukture podataka je hijerarhija graničnih opsega. Kod ovakvog tipa prostornih struktura podataka, particije se mogu presecati (slika 10.1). Drugi način organizacije primitiva podrazumeva particionisanje prostora u disjunktne regione, kao što su uniformna mreža ili kd stablo. U ovom pristupu primitive mogu biti sadržane u većem broju različitih regiona (slika 10.1). Ono što je dobra strana ovakvog vida particionisanja je to što je prvi određeni presek zraka i primitive zaista i globalno najbliži presek zraka i primitive.

U opštem slučaju ne postoji najbolja prostorna struktura podataka. Različite strukture podataka su pogodne za različite raspodele primitiva na sceni i različite upite. U daljem tekstu razmotrićemo različite interesantne prostorne strukture podataka i analizirati njihovu primenu na primeru izvršavanja često korišćenih upita.

10.1 Hijerarhija graničnih opsega

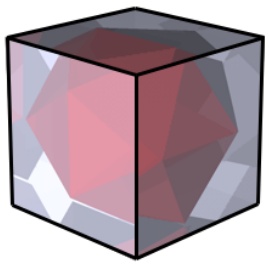
Granični opsezi (eng. bounding volume) predstavljaju jednostavne objekte koji se opisuju oko složenih objekata (slika 10.2). Motivacija za uvođenje graničnih opsega je ta što ako granični opseg nije vidljiv iz oka posmatrača, neće biti vidljiv nijedan objekat u njegovoj unutrašnjosti. Korišćenjem graničnih opsega može se ubrzati rej kasting algoritam. Naime, umesto da se vrši ispitivanje da li postoji presek zraka koji razmatramo u tekućem koraku sa složenim geometrijskim objektom, tj. sa svakim od poligona datog



Slika 10.1: Razlika između particionisanja primitiva i particionisanja prostora.

objekta, možemo ispitivati da li postoji presek zraka sa graničnim opsegom objekta. Ukoliko presek zraka i graničnog opsega ne postoji, iz daljeg razmatranja možemo izuzeti potencijalno veoma složeni objekat. Ukoliko presek zraka i graničnog opsega postoji, onda je moguće da zrak seče i neku primitivu iz graničnog opsega, te se mora ispitati presek sa svakom primitivom iz opsega. Stoga, složenost upita preseka u najgorem slučaju ostaje $O(N)$ gde je N broj primitiva.

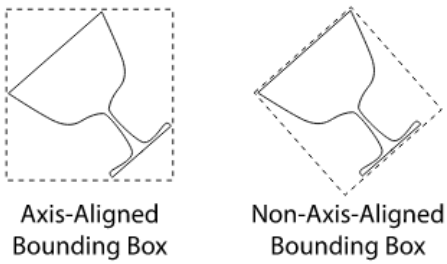
Izbor oblika graničnog opsega određen je sa dva donekle suprotstavljena cilja: s jedne strane, želeli bismo da koristimo granične opsege što jednostavnijeg oblika jer su računski efikasniji: zauzimaju manje memorije i ispitivanje preseka je jednostavnije. S druge strane, bilo bi dobro da granični opseg što tešnje odgovara objektu oko koga se opisuje, da bi se što ređe dešavala situacija da zrak preseca opseg, ali ne i sam objekat. U praksi se najčešće za granične opsege biraju kocke i sfere jer omogućavaju kompaktno čuvanje i efikasnije upite preseka.



Slika 10.2: Granični opseg u obliku kocke.

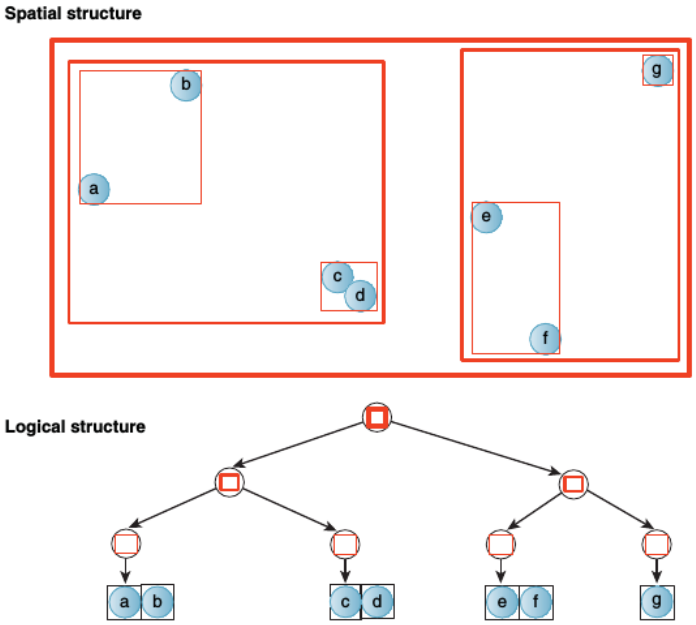
Granični opsezi poravnati sa koordinatnim osama koji se skraćeno nazivaju *AABB* (eng. Axis Aligned Bounding Boxes) (slika 10.3) su primer graničnog opsega koje je jednostavno čuvati i sa kojima je jednostavno izračunavati presek jer su poravnati sa koordinatnim osama. AABB imaju oblik kvadra i efikasno se konstruišu određivanjem minimuma i maksimuma x, y i z koordinata svih temena objekata (u trodimenzionom slučaju). Oni su i prostorno efikasni jer je za jedan opseg potrebno zapamtiti samo šest brojeva u pokretnom zarezu: x, y i z koordinatu donjeg levog i gornjeg desnog temena opsega. Korišćenje graničnih opsega poravnatih sa koordinatnim osama je pogodno za objekte koji imaju tesne granične opsege, dok nije pogodno ako postoje veliki prazni prostori unutar graničnog opsega jer će se često dešavati da zrak preseče granični opseg, a ne seče objekat unutar njega. AABB predstavljaju dobar izbor za dinamičke scene, jer je prilikom kretanja objekata, potrebno samo prilagoditi njihov granični opseg (računanjem minimuma i maksimuma) i izvršiti propagaciju naviše uz stablo. Iz tog razloga često se koriste za detekciju sudara (kolizije) u video igrama.

Rekurzivnim ugnježdavanjem graničnih opsega dobija se prostorno stablo poznato pod nazivom *hijerarhija graničnih opsega* (eng. Bounding Volume Hierarchy). Ovu strukturu podataka osmislili su Timoti Kaj (Timothy Kay) i Džejms Kajia (James Kajiya) 1986. godine. Na slici 10.4 prikazana je hijer-



Slika 10.3: Granični opseg poravnat sa koordinatnim osama dvodimenzionog objekta i granični opseg za koji to ne važi (preuzeto sa http://www.pbr-book.org/3ed-2018/Geometry_and_Transformations/Exercises.html).

arhija graničnih opsega u dvodimenzionom prostoru, u slučaju kada su granični opsezi pravougaonici sa stranicama poravnatim sa koordinatnim osama.



Slika 10.4: Hijerarhija dvodimenzionih graničnih opsega poravnatih sa koordinatnim osama.

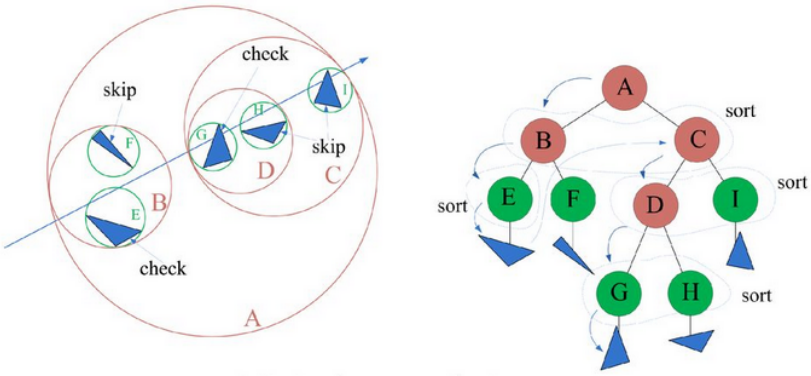
Za razliku od BSP stabala, ovaj tip stabla daje tešnje granice za klastere primitiva i ima konačnu zapreminu.

10.1.1 Konstrukcija hijerarhije graničnih opsega

Hijerarhija graničnih opsega se može graditi na dva načina: odozdo naviše i odozgo naniže. Konstrukcija odozdo naviše funkcioniše tako što se granični opsezi susednih objekata grupišu sve dok se kompletna scena ne ograniči jednim opsegom. U slučaju graničnih opsega poravnatih sa koordinatnim osama, grupisanje dva opsega se izvršava jednostavno, određivanjem minimuma i maksimuma opsega po svakoj koordinati. Međutim, određivanje objekata koji su blizu jedan drugom može biti težak problem. Naivno rešenje bi podrazumevalo poređenje svakog objekta sa svakim, što vodi kvadratnoj vremenskoj složenosti. Moguće je u ove svrhe iskoristiti graf scene i granični opseg u čvoru dobiti kao uniju graničnih opsega njegove dece, međutim graf scene oslikava pre svega logičku organizaciju scene, a ne nužno i prostornu.

Drugi način da se konstruiše hijerarhija graničnih opsega jeste odozgo naniže. U ovom pristupu krećemo od skupa objekata na sceni i rekurzivno vršimo podelu na dva podskupa (korišćenjem neke heuristike). Ovaj pristup se jednostavno implementira, ali daje stabla koja najčešće nemaju tako dobre performanse kao stabla koja su konstruisana tehnikom odozdo naviše.

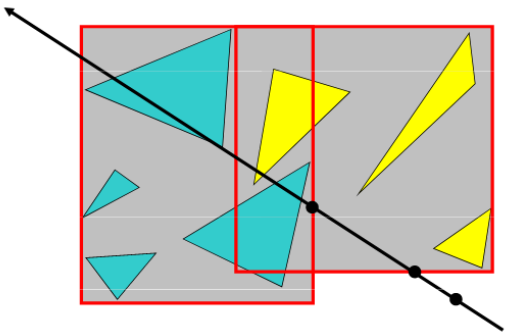
U hijerarhiji graničnih opsega se opsezi čvorova koji će biti objedinjeni zajedno nekim graničnim opsegom, nekada mogu preklapati (npr. primitive c i d sa slike 10.4).



Slika 10.5: Ilustracija upita preseka u slučaju hijerarhije graničnih opsega (granični opsezi su u obliku sfere).

10.1.2 Upit preseka za hijerarhiju graničnih opsega

Upit preseka se u slučaju hijerarhija graničnog opsega realizuje odozgo naniže, od korena ka listovima. Za tekući čvor stabla proverava se da li zrak seče tekući granični opseg: ukoliko presek ne postoji ili je presečna tačka dalja od trenutno najbližeg preseka, onda se vrši odsecanje podstabla sa korenom u datom čvoru, tj. ne ide se u opsege sadržane u datom. Ako zrak preseca tekući granični opseg, rekurzivno nastavljamo sa opsezima sadržanim u njemu. Pritom decu tekućeg čvora razmatramo u redosledu kojim je zrak sekao granične opsege, tj. kao prvi se razmatra onaj granični opseg koji je najbliži zraku (slika 10.5).



Slika 10.6: Situacija kada ne treba odmah vratiti pronađeni presek jer drugi potprostor ima bliži presek.

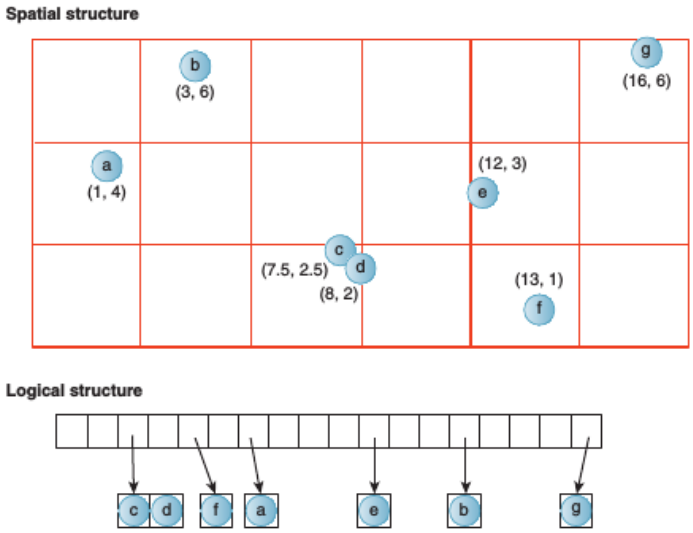
U slučaju da bratski opsezi imaju delove koji se preklapaju treba biti oprezan i ne vraćati odmah presek pronađen u prvom opsegu jer se može desiti da u drugom opsegu postoji bliži presek (slika 10.6). Očekivana vremenska složenost upita preseka je $O(\log N)$, za N primitiva na sceni, ali je u najgorem slučaju $O(N)$.

10.2 Uniformna mreža

Particionisanjem prostora na ćelije jednake veličine[1] dobija se *uniformna mreža* ili *rešetka* (eng. grid). Ova struktura podataka nije hijerarhijska, te zahteva više vremena za obilazak, ali se jednostavnije konstruiše. Takođe, za svaku tačka P u datom opsegu se jednostavno određuje u kojoj se ćeliji mreže nalazi: višedimenzioni indeks te ćelije se dobija oduzimanjem koordinata minimalnog temena mreže od koordinata tačke P, deljenjem sa dimenzijom jedne ćelije i zaokruživanjem naniže (slika 10.7). Stoga, ako se veličina ćelije održava konstantnom, ćelija kojoj primitiva pripada se može pronaći u konstantnom

[1] Ove ćelije su u literaturi poznate i kao vokseli.

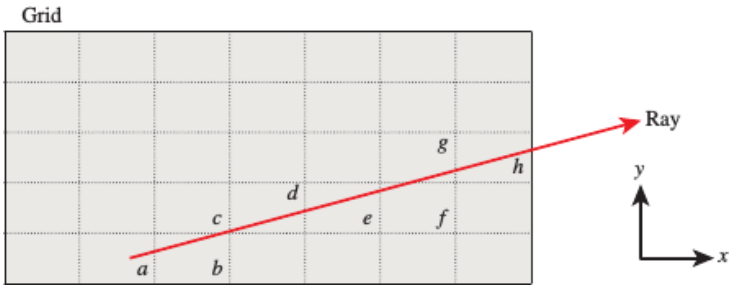
vremenu. Međutim, prostor potreban za skladištenje ove strukture podataka je proporcionalan sumi broja primitiva i zapremine mreže.



Slika 10.7: Dvodimenziona mreža koja čuva vrednosti sa pokazivačima na ključeve. Logička struktura odgovara jednodimenzionom nizu uređenom prema vrstama.

Uniformne mreže predstavljaju jednostavnu strukturu podataka, efikasno se konstruišu, a operacije dodavanja novih primitiva i brisanja primitiva nisu zahtevne ni vremenski ni prostorno i ne utiču na ostale primitive u mreži. Stoga su pogodne za dinamičke podatke i često se koriste za upite nalik onima kojima se ispituje da li je došlo do sudara dve primitive na sceni.

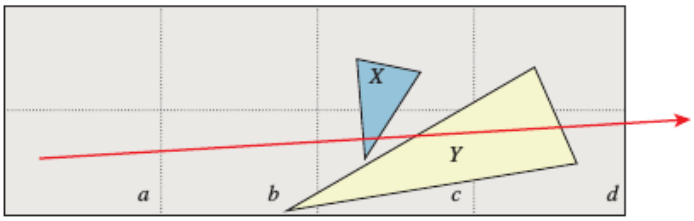
10.2.1 Upit preseka za uniformnu mrežu



Slika 10.8: Praćenje zraka kroz dvodimenzionu mrežu. Da bi se korektno pratio zrak, potrebno je posetiti ćelije uniformne mreže u redosledu a, b, c, d, e, f, g i h.

Kako bi se pronašao prvi presek zraka sa primitivom u uniformnoj mreži, algoritam za određivanje preseka mora da prođe kroz ćelije mreže u poretku u kom zrak ulazi u njih (slika 10.8). S obzirom na to da se primitive mogu protezati kroz više od jedne ćelije mreže, esencijalno je testirati samo preseke koji se javljaju u toj ćeliji u svakoj iteraciji. Ovaj princip je ilustrovan na slici 10.9. Posmatrajmo test preseka koji se javlja u iteraciji kada se nalazimo u ćeliji označenoj sa b. S obzirom na to da ćelije koje pokriva objekat Y uključuju ćeliju b, tokom ove iteracije testiraćemo Y u odnosu na zrak. Presek postoji, ali se ne nalazi u ovoj ćeliji već u ćeliji c. Ukoliko bi bio vraćen taj presek, ne bismo dobili pravi prvi presek zraka i primitive na sceni koji se javlja između objekta X i zraka u ćeliji c.

Upit preseka se izvršava u vremenu proporcionalnom broju ćelija uniformne mreže kojima zrak prolazi. Cena svake iteracije je konstantna, te je algoritam od praktične koristi u slučajevima kada ne očekujemo da zrak putuje predugo pre nego što "udari" u prvi objekat. Mreža sa g podela duž k dimenzija sadrži $O(g^k)$ ćelija. Najduži put zraka je dijagonala mreže dužine $O(g)$. Za mrežu koja sadrži n primitiva, u najgorem slučaju sve primitive se protežu kroz sve ćelije mreže duž dijagonale, a zrak ne preseca nijednu od njih. Cena upita preseka bi stoga u najgorem slučaju bila $O(g \cdot n)$. U ovakvoj



Slika 10.9: Slučaj kada je važno testirati samo presek koji se nalazi u ćeliji u kojoj se trenutno nalazimo.

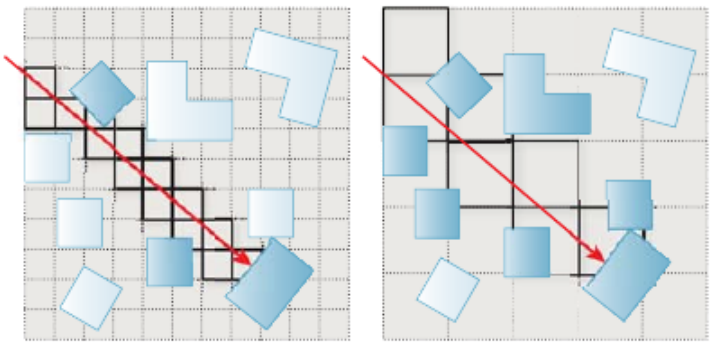
situaciji mreža predstavlja loš izbor prostorne strukture podataka. Dakle, ponašanje u najgorem slučaju je veoma različito od očekivanog ponašanja.

Uniformna mreža je pogodnija za scenu koja sadrži približno ravnomerno raspodeljene primitive koje teže tome da stanu u ćeliju mreže. Na upitima za velike scene sa neravnomernom prostornom raspodelom primitiva, stabla pokazuju bolje asimptotsko ponašanje od mreža: kod stabla očekujemo logaritamsko vreme izvršavanja, dok je kod mreža očekivano vreme izvršavanja linearno. Međutim, kroz praznu ćeliju se može brzo proći. Stoga, ako linearan broj puta primenimo operaciju konstantne cene, možemo priuštiti prolazak velikog broja ćelija za svaki zrak. Za scene sa ograničenom veličinom, mreža može da pokaže bolje performanse od stabla za upite preseka, posebno ako je vreme potrebno za izgradnju strukture podataka uključeno u vreme izvršavanja.

10.2.2 Konstrukcija uniformne mreže

Kao što smo već pomenuli, konstrukcija uniformne mreže je veoma jednostavna. Međutim, važno pitanje prilikom konstrukcije jeste odabir rezolucije mreže, odnosno veličine ćelije mreže. Podešavanje veličine ćelije uniformne mreže, odnosno broja ćelija za dati opseg mreže zahteva dobro poznavanje tipova upita koji će biti izvršavani, kao i poznavanje prostorne raspodele podataka. Razmotrimo na koji način izabrati rezoluciju mreže, tako da mreža ima što bolje performanse. Asimptotska složenost određivanja preseka zraka na mreži je linearna po broju ćelija mreže koje zrak preseca. Dakle, što manje ćelija zrak preseca, manja će biti i složenost određivanja preseka. Upit preseka ima i linearnu složenost u funkciji broja primitiva koje se nalaze u mreži. Dakle, da bismo dobili efikasniji algoritam, mogu se razmatrati dva različita cilja a to su: minimizovati broj testiranja preseka sa mrežom i minimizovati broj testiranja preseka sa primitivama. Neka razumna pretpostavka je da je utvrđivanje preseka sa mrežom "jeftinije" od utvrđivanja preseka sa primitivom.

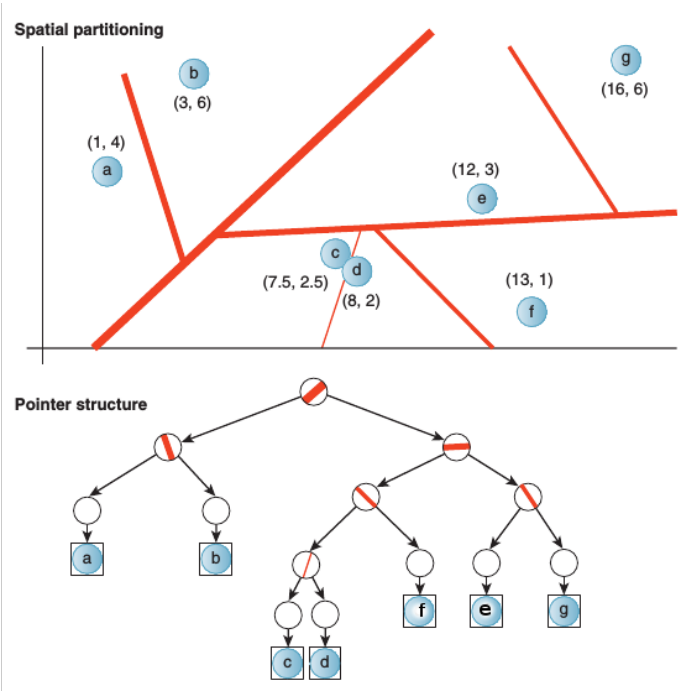
Ako je broj podela g veliki, kvadrati mreže su mali. Ovim se smanjuje broj primitiva koje treba testirati u svakoj nepraznoj ćeliji, ali se povećava broj ćelija koje treba ispitati. Ovaj izbor je dobar za guste scene. Ako je pak vrednost g mala, kvadrati mreže su veliki. Kod ovakvih mreža zrak se brzo kreće kroz velike prazne prostore, ali se povećava broj primitiva u svakoj nepraznoj ćeliji sa kojom treba vršiti testiranje. Manje vrednosti za g su pogodnije za retke scene (slika 10.10).



Slika 10.10: Na slici levo ilustrovana je mreža sa velikim brojem podela: zrak treba da prođe kroz mnogo ćelija ali će mnoge od njih biti prazne. Na slici desno ilustrovan je slučaj uniformne mreže sa malim brojem podela: zrak iterira kroz mali broj ćelija, ali one mogu da sadrže veliki broj primitiva.

10.3 Binarno stablo prostornog particionisanja

Koncept *binarnih stabala prostornog particionisanja* (skraćeno BSP stabala[2]) su 1980. godine predložili Henri Fuks (Henry Fuchs), Zvi Kedem (Zvi Kedem) i Brus Nejlor (Bruce Naylor). U slučaju dvodimenzionog BSP stabla ravan se deli pravama razdvajanja, dok se u slučaju trodimenzionog BSP stabla trodimenzioni prostor deli ravnima razdvajanja. Na slici 10.11 data je ilustracija dvodimenzionog BSP stabla.



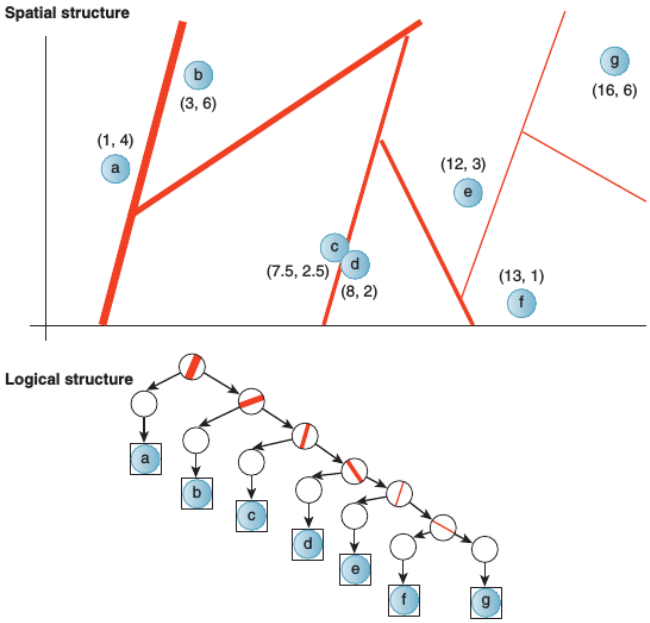
Slika 10.11: Ilustracija dvodimenzionog BSP stabla. Debljina prave razdvajanja odgovara dubini odgovarajućeg čvora u stablu; koren je predstavljen najdebljom linijom.

BSP stablo se dobija rekurzivnim binarnim particionisanjem prostora. Ravni razdvajanja dele prostor u konveksne potprostore. Preciznije, listovi BSP stabla odgovaraju potprostorima, a unutrašnji čvorovi ravnima razdvajanja. Unutrašnjim čvorovima odgovaraju i konveksni prostori koji su unija njihove dece. Iako svi listovi predstavljaju konveksne prostore, oni koji odgovaraju prostornim ekstremumima predstavljaju prostore beskonačne zapremine (npr. potprostor koji sadrži primitivu g na slici 10.11). Iako prostor beskonačne zapremine može delovati nepogodno za neka izračunavanja, na ovo svojstvo BSP stabla se može gledati i kao na njegovu prednost jer se BSP stablom predstavlja čitav prostor. Naime, i bez izmene strukture stabla moguće je dinamički dodavati i uklanjati primitive, dok se kod prostornih struktura podataka koje predstavljaju samo konačnu zapreminu (površinu) sama struktura podataka mora izmeniti ako se primitiva pomeri izvan prethodnih granica scene (npr. kod uniformne mreže).

10.3.1 Konstrukcija BSP stabla

U nekim situacijama može se desiti da dodavanjem i brisanjem primitiva dobijemo jako neuravnoteženo stablo u kome većina čvorova ima samo jedno dete (slika 10.12). Ova situacija se može desiti čak i ako imamo dobru strategiju izgradnje stabla, ako se nakon postavljanja ravni razdvajanja primitive pomeraju. Naime, prilikom pomeranja pritimiva može se desiti da neke particije postanu prazne i da stablo bude dosta veće veličine nego što je to potrebno. Napomenimo da se nekada ne postavlja gornja granica za veličinu stabla. Jedan od razloga za to je taj što se particije mogu dodavati nezavisno od primitiva u slučaju da očekujemo da nove primitive budu naknadno dodate. Ipak u većini algoritama ćemo podrazumevati da je stablo izgrađeno pod nekim razumnim pretpostavkama i da u slučaju kad sadrži n primitiva ima manje od $2n$ čvorova. U balansiranom BSP stablu moguće je locirati list korišćenjem $O(\log n)$ poređenja, dok se ovo vreme može povećati na $O(n)$ ako stablo nije balansirano. Vreme izvršavanja upita preseka raste sa porastom veličine prostora koji je pokriven geometrijom upita.

[2]Ovu skraćenicu ne treba mešati sa binarnim stablima pretrage, odnosno uređenim binarnim stablima.



Slika 10.12: Degenerisano BSP stablo sa istim asimptotskim ponašanjem kao i lista. Loše performanse stabla potiču od neefikasnog izbora particija.

Srećom ovaj rast je uglavnom u vremenu manjem od linearnog. Osnovna pretpostavka je sledeća: ako zrak pređe veliko rastojanje pre nego što naiđe na primitivu, on će najverovatnije proći kroz veliki broj particija, a ako pređe kratak put proći će kroz mali broj njih. Očekujemo da kad primitive imaju kolikotoliko uniformnu raspodelu, particije formiraju balansirano stablo i da će vreme izvršavanja upita preseka rasti logaritamski sa opsegom geometrije upita.

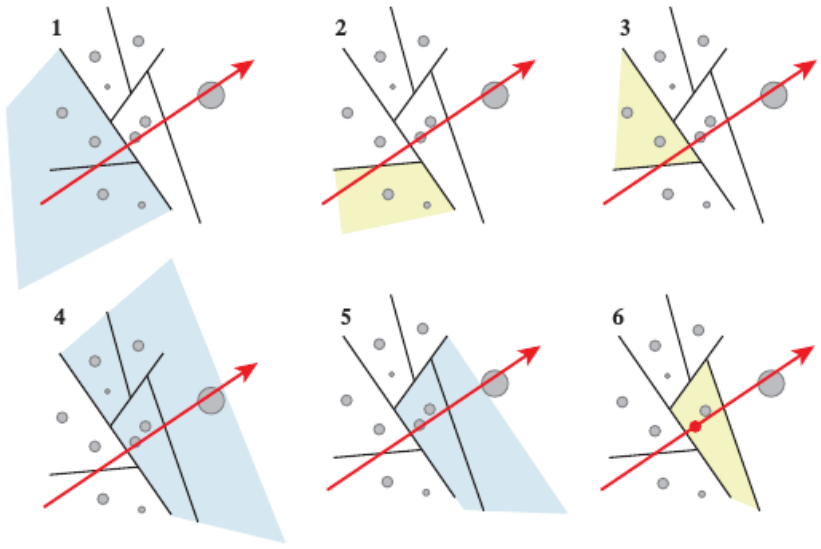
Balansiranost BSP stabla, odnosno dubina stabla, je važna za primene poput rej kasting algoritma, kod koga se tipično prate nekoliko putanja od korena do lista.. Međutim, za primene poput odbacivanja skrivenih površi, kod kojih je potrebno obići kompletno stablo, balansiranost stabla ne utiče značajno na vreme izvršavanja jer je vreme obilaska stabla linearno u funkciji broja čvorova u stablu i ne zavisi od dubine stabla.

Dakle, efikasnost rada sa BSP stablom zavisi od izbora ravni razdvajanja. Međutim, izbor dobrih ravni razdvajanja nije jednostavan; šta više, konstrukcija optimalnog BSP stabla je NP-kompletan problem. Teorijski, postoji beskonačno mnogo ravni između kojih treba napraviti izbor. Pritom sam izbor ravni razdvajanja zavisi od vrste upita koji želimo da optimizujemo, od raspodele podataka i od toga da li želimo da optimizujemo najgori slučaj, najbolji slučaj ili prosečni slučaj. Ako se stablo gradi samo jednom, na početku, onda se može dozvoliti da algoritam konstrukcije stabla bude i veće vremenske složenosti. Ukoliko se stablo gradi tokom rada programa ili će tokom izvršavanja biti često menjano, važno je minimizovati vreme koje zajedno troše izgradnja stabla i izvršavanje upita. U nekim situacijama se ravni razdvajanja biraju iz skupa strana ulaznog skupa poligona, nekad je pak najbolje birati ravni razdvajanja tako da budu ortogonalne na koordinatne ose. Dodatno, nekada se algoritam konstrukcije stabla zaustavlja tek kad se svaka primitiva razdovoji od preostalih (svaki list sadrži po jednu primitivu), a nekada se algoritam zaustavlja ranije – kada broj primitiva u svakom od listova postane dovoljno mali ili se dostigne maksimalna dozvoljena dubina stabla.

Ukoliko ravan razdvajanja preseca neku primitivu sa scene, onda se ta primitiva deli na dve primitive u odnosu na datu ravan razdvajanja. Dakle, prilikom konstrukcije BSP stabla se može desiti da zbog podele primitiva stablo sadrži dosta veći broj poligona nego inicijalna scena. Naravno, što je veći broj poligona, to je potrebno i više memorije za njihovo skladištenje i operacije nad odgovarajućim BSP stablom duže traju. Dakle, bilo bi dobro minimizovati broj podela odabirom pogodnih ravni razdvajanja, ali to pretpostavlja neko predznanje o poligonima koji će biti dodavani u stablo, što u interaktivnim aplikacijama često nije moguće.

10.3.2 Upit preseka za BSP stablo

U situaciji kada su primitive na sceni organizovane korišćenjem BSP stabla upit preseka se realizuje odozgo naniže, od korena ka listovima. Za tekući čvor se najpre proverava da li je u pitanju list – ako jeste, proverava se da li postoji presek sa nekom od primitiva sadržanom u listu. Inače se najpre ide u dete koje je geometrijski bliže – ako se pronađe presek u bližem detetu, pretraga se prekida i ne ide se u dalje dete. Algoritam utvrđivanja prvog preseka ilustrovan je na slici 10.13.



Slika 10.13: Praćenje zraka kroz scenu koja se čuva u dvodimenzionom BSP stablu. Osenčeni deo prostora odgovara čvoru nad kojim algoritam radi u tekućem koraku. Iteracija napreduje u dubinu, s tim da se prvo obilazi bliže dete tekućeg čvora.

U najgorem slučaju algoritam mora da poseti svaki čvor stabla, međutim, u praksi se ovo retko događa. Obično se do prvog preseka pređe put koji je dosta manji od veličine scene. Stoga je očekivano dobiti relativno usku pretragu u dubinu sa vremenom izvršavanja proporcionalnom visini stabla, odnosno očekivana složenost upita preseka za BSP stablo je $O(\log N)$. Stoga se korišćenjem ove strukture podataka rej kasting algoritam može učiniti upotebljivijim za složene scene.

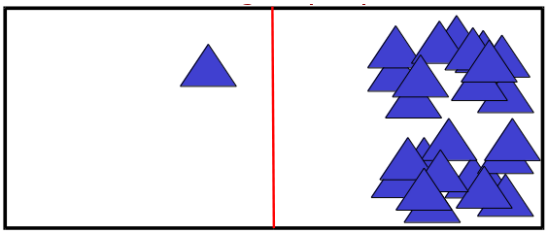
10.4 kd stabla

BSP stablo kod koga su ravni razdvajanja upravne na koordinatne ose naziva se *kd stablo* (eng. kd tree)[3]. Ovu strukturu je osmislio Džon Bentli (Jon Bentley) 1970. godine. Koordinatna osa na koju je ravan razdvajanja upravna bira se na osnovu raspodele samih podataka ili u cikličnom poretku (npr. u slučaju dimenzije 3, koren bi odgovarao ravni razdvajanja upravnoj na x osu, njegova deca ravnima upravnim na y osu, unuci korena stabla ravnima upravnim na z osu, njihova deca ravnima upravnim na x osu itd). Na ovaj način se i smanjuje broj bitova koji je potreban za predstavljanje particija, čime se smanjuje cena skladištenja same strukture. Bez obzira na postavljena ograničenja na ravni razdvajanja, i dalje je broj mogućih načina na koje se one mogu birati ogroman.

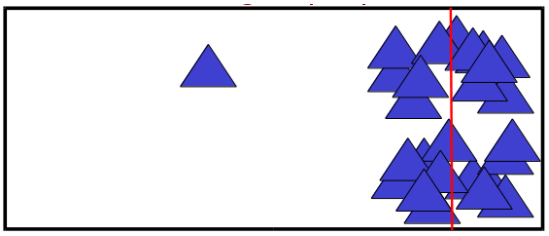
10.4.1 Konstrukcija kd stabla

Postoje različite strategije konstrukcije kd stabala. Neke od mogućih strategija odabira ravni razdvajanja su da se u svakom od podstabala izvrši podela po sredini opsega (slika 10.14) ili po vrednosti medijane primitiva (slika 10.15). Kada se podela vrši po sredini opsega, levi i desni potprostor biće jednake zapremine, te zrak sa jednakom verovatnoćom može da uđe u levi i desni potprostor. Međutim, ako zrak preseče desni potprostor, moraće da se izvrši test preseka sa skoro svakim trouglom. U slučaju kada se podela vrši po vrednosti medijane pozicija primitiva, ukoliko zrak uđe u bilo koji od regiona zahteva ispitivanje da li postoji presek sa bar polovinom trouglova sa scene.

[3]Originalno je bilo zamišljeno da se vrednost k u nazivu strukture podataka menja dimenzijom prostora, pa da tako recimo govorimo o 2d stablima ili 3d stablima, međutim ustalio se izraz kd stablo dimenzije d.

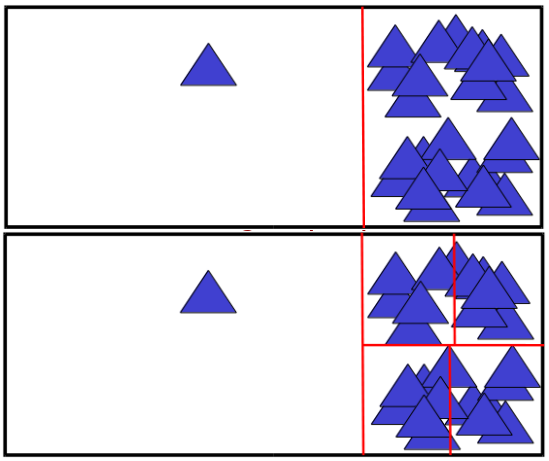


Slika 10.14: Podela po sredini opsega.



Slika 10.15: Podela po vrednosti medijane pozicija primitiva.

Pored navedenih strategija, moguće je ravni razdvajanja birati tako da se poveća verovatnoća da je potrebno testirati samo jedan trougao maksimizovanjem površina čvorova koje sadrže malo primitiva. U slučaju kada su primitive raspoređene kao na slici 10.15, u prvom koraku se može izdvojiti veliki prostor koji sadrži samo jednu primitivu i koji ne zahteva dodatnu podelu, dok se drugi potprostor može dalje deliti bilo po sredini opsega, bilo prema medijani (slika 10.16). Na ovaj način kreiraju se veliki prazni čvorovi koji se tokom obilaska mogu brzo odbaciti. Pokazuje se da su najuspešnije strategije one koje maksimizuju prazne prostore u što ranijim koracima podele.



Slika 10.16: Prikaz prva dva koraka podele u skladu sa optimalnom cenom; levo dete u prvom koraku ne zahteva dalje podele, a desno dete (i svaki od njegovih potomaka) se deli prema vrednosti medijane.

Pretpostavimo da su zraci ravnomerno raspodeljeni u prostoru. Broj mogućih pozicija ravni razdvajanja je beskonačan, te kao moguće ravni razdvajanja razmatramo samo podele duž koordinatnih osa koje sadrže ivice primitiva. Cenu neke ravni razdvajanja računamo po formuli:

$$c = P_l \cdot c_l + P_d \cdot c_d$$

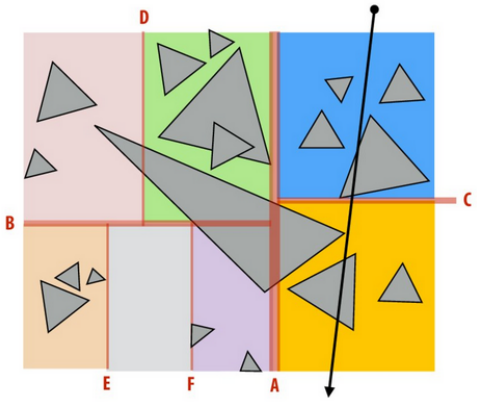
gde je sa P_l i P_d označen količnik površina[4] graničnih opsega roditeljskog čvora i levog, odnosno desnog deteta (najčešće graničnog opsega poravnatog sa koordinatnim osama), a sa c_l i c_d broj primitiva koje sadrži levi, odnosno desni potprostor. Od svih razmatranih ravni razdvajanja u tekućem čvoru biramo

[4]U trodimenzionom prostoru razmatramo površinu graničnih opsega, dok kad radimo u dvodimenzionom prostoru razmatramo obim graničnog opsega. Naime zrak u 2D opseg ulazi kroz neku tačku sa obima opsega, a u 3D opseg kroz neku tačku sa površine opsega, te je na taj način procenjena verovatnoća da zrak uđe u opseg. Dodatno, razmatranje zapremine opsega u trodimenzionom prostoru (odnosno površine u slučaju ravni) ne bi funkcionisalo jer bi se u nekim situacijama dešavalo da cena nakon podele bude uvek manja od cene ulaska u celi opseg bez dalje podele što bi navelo da vršimo podele beskonačan broj puta.

onu koja ima najmanju cenu. Podelu vršimo samo ako je cena nakon podele manja od cene ulaska u tekući čvor bez dalje podele. Ova heuristika prati pohlepnu strategiju i naziva se *heuristika površine površi* (eng. surface area heuristic). Primetimo na kraju da su za izbor ravni razdvajanja sa slike 10.14 vrednosti P_l i P_d jednake, dok su u slučaju odabira ravni razdvajanja prikazanog na slici 10.15 vrednosti c_l i c_d jednake. Odabir ravni razdvajanja prikazan na slici 10.16 odgovara podeli sa najmanjom cenom.

10.4.2 Upit preseka za kd stablo

Upit preseka za kd stablo se realizuje odozgo naniže, od korena ka listovima. Za tekući čvor se najpre proverava da li je u pitanju list – ako jeste, proverava se da li postoji presek sa nekom od primitiva sadržanom u listu. Ako čvor nije list pretraga se nastavlja u bližem detetu – onom za koji važi da je prvi presek zraka i opsega bliže početku zraka. Ako ne postoji presek zraka i graničnog opsega, ne ide se u decu tekućeg čvora jer neće postojati ni presek sa njima. Prilikom nailaska na prvi presek zraka i primitive, pretraga se prekida (slika 10.17).



Slika 10.17: Ilustracija upita preseka za kd stablo.

10.5 Oktri

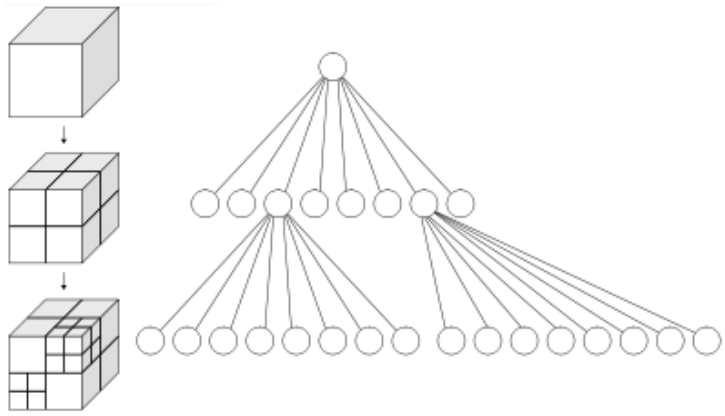
Razmotrimo kocku koja sadrži celu scenu. Podela kroz središte kocke duž svih k koordinatnih osa daje skup ugnježdenih k-kocki[5] (u dvodimenzionom slučaju dobijaju se kvadrati, a u trodimenzionom slučaju kocke). Ova struktura podataka se u slučaju trodimenzionog prostora naziva *oktri* (eng. oct tree) i prikazana je na slici 10.18. Kocka se deli sa tri ravni kroz središte opsega: jedna kocka deli se na osam manjih kocki. Ova struktura se najčešće predstavlja u vidu stabla u kom svaki unutrašnji čvor ima osmoro dece. Za oktri važi da nisu nužno sve kocke iste veličine, već se gušći delovi scene sastoje od većeg broja manjih kocki, dakle struktura stabla se prilagođava sadržaju scene. Za razliku od BSP stabala, svi čvorovi oktrija odgovaraju prostorima konačne zapremine.

Struktura podataka analogna oktriju u ravni je *kuodtri* (eng. quad tree), kod koga se svaki pravougaonik rekurzivno deli na 4 manja pravougaonika. Na slici 10.19 prikazana je struktura podataka kuodtri u slučaju scene prikazane na slici 10.11. U ovom primeru sve podele nakon prve biće veoma slične podelama koje se vrše u slučaju kd stabala. Razlika je jedino u tome što je u slučaju kd stabla već nakon prve podele jedna primitiva razdvojena od preostale geometrije, dok bi se u slučaju strukture podataka kuodtri razdvajanje primitiva od ostalih dogodilo nešto kasnije (dobija se stablo veće dubine koje je manje efikasno pretraživati).

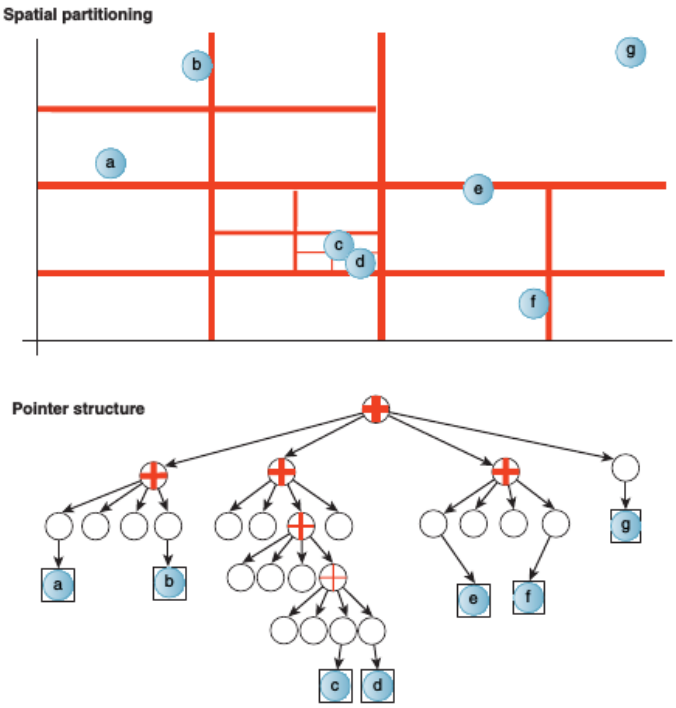
10.5.1 Konstrukcija oktrija

Oktri se konstruišu odozgo naniže: najpre se pronalazi kocka koja ograničava celu scenu – ona odgovara korenu stabla koje sadrži sve primitive. U svakoj iteraciji, tekući čvor particionišemo na osam podudarnih oktanata i delimo primitive u oktante: ako se neka primitiva prostire u više oktanata, dodeljujemo je u oba. Primetimo da ovakva konstrukcija ne garantuje da će dobijeno stablo biti balansirano. Iz tog razloga postavljamo ograničenje na maksimalnu dozvoljenu dubinu oktrija. Nastavljamo

[5]Nije nužno da potprostori budu u obliku kocke, već mogu biti i u obliku kvadra. Zbog pogodnosti u daljem tekstu koristićemo termin kocka.



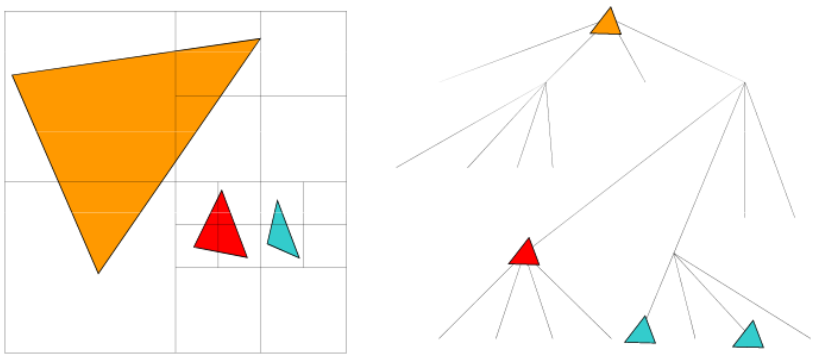
Slika 10.18: Ilustracija strukture podataka oktri. Na levoj strani prikazana je rekurzivna podela kocke na 8 kocki, a desno odgovarajuća drvolika struktura.



Slika 10.19: Ilustracija strukture podataka kuodtri. Zbog blizine primitiva c i d, stablo je velike dubine. U ovom slučaju bi se više isplatilo da napravimo čvor drugog nivoa koji sadrži ove dve primitive, umesto da vršimo podele sve dok svaki list stabla ne sadrži samo jednu primitivu.

rekurzivno sve dok ne dođemo do maksimalne dozvoljene dubine stabla ili do situacije kada oktant sadrži dovoljno malo primitiva. Varijantu oktrija kod koje se primitive čuvaju samo u listovima stabla nazivamo *oktri sa duplikatima* jer ako jedna ista primitiva pripada većem broju oktanata, ona će se veći broj puta pamtiti u strukturi. U slučaju većih objekata koji pokrivaju veliki broj oktanata, podoktanata itd., objekat bi trebalo da se pamti u velikom broju listova, čime se povećavaju i prostorna i vremenska složenost rada sa ovom strukturom podataka. U ovoj situaciji pogodnije je da se objekti ne ubacuju u podoktante tekućeg čvora već da se pamte u tom unutrašnjem čvoru. Ovu vrstu oktrija nazivamo *oktri bez duplikata* (slika 10.20). Napomenimo da i za ovu vrstu oktrija postoji scenario kada iskazuju loše ponašanje: kada bi svi objekti bili sadržani u većem broju oktanata polaznog čvora (korena), svi objekti bi bili sačuvani u korenu čime bi se izgubile prednosti hijerarhijske strukture ove strukture podataka.

Za scene kod kojih jedan deo scene sadrži veliki broj primitiva dok je drugi gotovo prazan, oktri će imati veoma loše performanse jer se generiše veći broj podela za deo scene koji sadrži veliki broj objekata. Na ovaj način se generišu stabla velike dubine koja je neefikasno obići. U praksi, nažalost, ovakve scene nisu tako neuobičajene. Da bi se ovaj problem donekle smanjio moguće je zadati maksi-

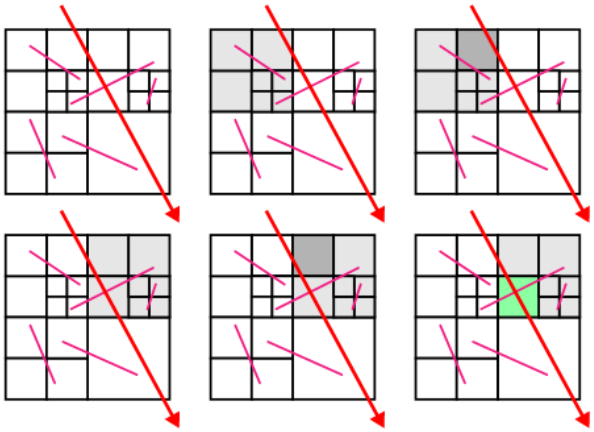


Slika 10.20: Oktri bez duplikata, u kom je moguće čuvati primitive i u unutrašnjim čvorovima stabla.

malnu dozvoljenu dubinu stabla.

10.5.2 Upit preseka za oktri

Razmotrimo problem rešavanja upita preseka u slučaju strukture podataka oktri. Potrebno je najpre rešiti jedan jednostavan problem a to je pronalaženje onog preseka kocke i zraka koji je najbliži početku zraka. Kocka ima šest strana i potrebno je naći najbliži presek, ako uopšte postoji. Potraga za prvim presekom zraka i primitive počinje u korenu. Ako je čvor list, za svaku od primitiva u tom čvoru potrebno je ispitati da li postoji presek sa zrakom. Inače decu tog čvora sortiramo rastuće prema rastojanju početka zraka od prvog preseka kocke i u tom redosledu iteriramo kroz njegovu decu. Ispitujemo da li postoji presek zraka i kocke koja odgovara detetu: ako presek postoji, nastavljamo rekurzivno na tom detetu, a ako presek ne postoji vršimo odsecanje pretrage u podstablu sa korenom u tom detetu. Kada se pronađe prvi presek, dalja pretraga se prekida (slika 10.21).



Slika 10.21: Ilustracija izvršavanja upita preseka u slučaju strukture podataka oktri. Sivom bojom označen je kvadrat za koji se trenutno vrši ispitivanje preseka i to svetlo sivom roditeljski čvor, a tamno sivom list. Zelenom bojom označen je list u kome je pronađen prvi presek zraka i primitive sa scene.

Napomenimo na kraju da se različite prostorne strukture podataka mogu i međusobno kombinovati: na primer, da se na najvišem nivou koristi uniformna mreža, a da se objekti unutar ćelije mreže organizuju korišćenjem hijerarhija graničnih opsega.

Na kraju, podvucimo još jednom da izbor pogodne prostorne strukture podataka zavisi od broja primitiva, zapremene scene koju zauzimaju i njihove raspodele na sceni. Svakako, bez obzira na odabranu strukturu podataka, važno je razumeti njene prednosti i ograničenja kako bi se obezbedio pravi izbor.

10.6 Pitanja

10.1 Koje figure se najčešće koriste kao granični opsezi?

10.2 Da li je AABB specijalan slučaj (a) hijerarhije graničnih opsega (b) BSP stabla (c) oktrija?

10.3 Kako se realizuje upit preseka kod hijerarhije graničnih opsega?

10.4 Kolika je očekivana vremenska složenost upita preseka, a kolika vremenska složenost u najgorem slučaju ako koristimo uniformnu mrežu kao prostornu strukturu podataka?

10.5 Koje su prednosti a koji nedostaci uniformne mreže kao prostorne strukture podataka?

10.6 Uporediti performanse uniformne mreže sa malom veličinom ćelije sa mrežama kod kojih je veličina ćelije velika.

10.7 Kolika je očekivana vremenska složenost upita preseka, a kolika vremenska složenost u najgorem slučaju ako se koristi BSP stablo?

10.8 Kako se biraju particije kod kd stabla? Kako se najčešće biraju ravni razdvajanja kod kd stabala?

10.9 Koja je razlika između binarnog stabla prostornog particionisanja i kd stabla?

10.10 Koja je razlika između korišćenja BSP stabala i hijerarhija graničnih opsega?

10.11 Koja je struktura podataka pogodnija za dinamičke podatke: BSP stablo ili uniformna mreža?

10.12 Kako glasi heuristika površine površi?

10.13 Kako izgleda struktura podataka oktri, a kako kuodtri?

10.14 Na koji način se obrađuje situacija kada se neka primitiva proteže kroz više ćelija oktrija?

10.15 Na koji način se konstruiše oktri za datu scenu?

10.16 Kako se realizuje upit preseka kod strukture podataka oktri?

10.17 Prilikom rešavanja upita preseka korišćenjem oktrija, u kojoj situaciji možemo prekinuti pretragu u podstablu sa korenom u tekućem čvoru, a u kojoj situaciji moramo nastaviti pretragu u deci tekućeg čvora?

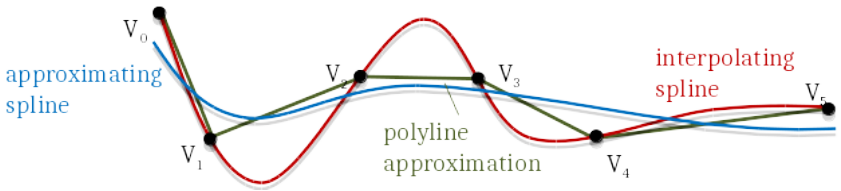
Glava 11

Zadavanje krivih i površi

Glatke krive i površi su neophodne u mnogim primenama računarske grafike: na primer, kretanje kamere je poželjno opisati glatkom krivom, a neke objekte modelovati glatkim površima. U ovom poglavlju razmatraćemo koji su najbolji načini za predstavljanje krivih i površi u trodimenzionom prostoru, tako da se objekti sa kojima se uobičajeno radi mogu jednostavno predstaviti, da je lako interagovati sa njima (npr. prilagoditi ih nekim merama) i da su podržana osnovna geometrijska izračunavanja, poput upita preseka, detekcije kolizije i sl.

11.1 Predstavljanje krivih u trodimenzionom prostoru

Razmotrimo najpre rad sa krivama u trodimenzionom prostoru. Jedno od prvih pitanja koje se postavlja jeste koji je najbolji način da zapamtimo krivu na računaru kako bi to bilo pogodno za primene u računarskoj grafici. Krive nije pogodno čuvati u rasterskom obliku jer je to memorijski zahtevno, a, pritom, ovakva reprezentacija ima i dosta drugih nedostataka kao što je alijasing efekat i nemogućnost skaliranja bez gubitka na kvalitetu. Dakle, potrebna nam je neka efikasnija matematička reprezentacija. Možemo pamtiti listu kontrolnih tačaka, a između njih vršiti neki vid interpolacije. Korišćenje trigonometrijskih funkcija za interpolaciju je skupo i njima se teško manipuliše. Deo po deo linearna aproksimacija između kontrolnih tačaka nije glatka i ukoliko broj kontrolnih tačaka nije dovoljno veliki ovaj vid interpolacije neće dati vizuelno zadovoljavajući izgled. Međutim, polinomi stepena većeg od 1 se relativno jeftino računaju i njima se jednostavno manipuliše (na primer, izvod se računa veoma jednostavno), te oni predstavljaju pogodan izbor za interpolaciju između kontrolnih tačaka. Pritom se najčešće koriste polinomi trećeg stepena jer je 3 najmanji stepen parametarske krive takav da kriva ne mora nužno da pripada ravni. Zaista, kriva stepena 2 je u potpunosti određena trima tačkama, a te tri tačke određuju ravan (kojoj pripada ta kriva).



Slika 11.1: Različiti načini zadavanja krive koja je vođena kontrolnim tačkama: deo po deo linearna interpolacija, interpolišući i aproksimirajući splajn.

Krive se mogu opisati na različite načine:

eksplicitnom reprezentacijom: $y = f(x), z = g(x)$

potencijalni problem ove reprezentacije je taj što za jednu vrednost x postoji samo jedna tačka na krivoj, te je u nekim slučajevima potrebno kombinovati više krivih; na primer pri modelovanju sfere

implicitnom reprezentacijom: $f(x, y, z) = 0$

potencijalni problem ove reprezentacije je što može da dâ više rešenja nego što nam treba; na primer, za modelovanje sfere možemo da koristimo jednačinu $x^2 + y^2 + z^2 = 1$, ali da bismo

modelovali samo gornji deo sfere potrebno je koristiti dodatne uslove, što može stvoriti druge probleme.

parametarskom reprezentacijom: $x = x(t), y = y(t), z = z(t),\ 0 \leq t \leq 1$

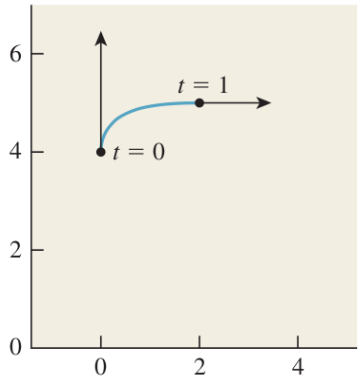
ova reprezentacija nema većinu problema koje imaju prethodne dve. Za funkcije x, y i z se najčešće koriste kubni polinomi.

11.2 Osnovne polinomijalne krive

Kriva trećeg stepena $\gamma(t) = a_3t^3+a_2t^2+a_1t+a_0$ zadaje se vrednostima njena četiri koeficijenta. Da bismo odredili vrednosti ovih koeficijenata, potrebno je zadati neka četiri ograničenja na krivoj: to mogu biti četiri tačke koje kriva sadrži ili tačke kojima se vrši navođenje krive, a u igru mogu ući i vektori tangenti krive u nekim tačkama. U ovom poglavlju razmotrićemo Hermitovu i Bezjeovu krivu kao primere često korišćenih kubnih krivih.

11.2.1 Hermitova kriva

Razmotrimo naredni problem. Automobil se kreće duž y ose sa vektorom brzine $[0, 3]^T$ i stiže u tačku $(0, 4)$ u trenutku $t = 0$. Potrebno je modelovati kretanje automobila koje uključuje skretanje i usporavanje tako da automobil u trenutku $t = 1$ stigne u poziciju $(2, 5)$ sa vektorom brzine $(2, 0)$ (slika 11.2).



Slika 11.2: Hermitova kriva.

Potreban nam je način da "zalepimo" dva dela putanje automobila kako bismo dobili glatko kretanje: treba povezati kretanje duž y ose do tačke $(0, 4)$ sa kretanjem duž prave $y = 5$ od tačke $(2, 5)$.

Najpre ćemo uopštiti problem. Za date pozicije P i Q i vektore brzina v i w potrebno je odrediti funkciju $\gamma : [0, 1] \to R^2$ tako da je $\gamma(0) = P$, $\gamma(1) = Q$, $\gamma'(0) = v$ i $\gamma'(1) = w$. Rešenje se dobija prostom zamenom gornja četiri uslova u opšti oblik kubnog polinoma $\gamma(t) = a_3t^3 + a_2t^2 + a_1t + a_0$ i ima oblik:

$$\begin{aligned}\gamma(t) &= (2t^3 - 3t^2 + 1)P + (-2t^3 + 3t^2)Q + (t^3 - 2t^2 + t)v + (t^3 - t^2)w \\ &= (1 - t)^2(2t + 1)P + t^2(-2t + 3)Q + t(t - 1)^2v + t^2(t - 1)w.\end{aligned}$$

Rezultujuća kriva naziva se *Hermitova kriva* (eng. Hermite curve) za ulaz P, Q, v i w. Ime je dobila po francuskom matematičaru Čarlsu Hermitu (Charles Hermite). Četiri polinoma u gornjoj jednačini nazivaju se *Hermitovim baznim funkcijama* i oni čine *Hermitovu bazu*.

Ukoliko želimo da samo promenimo početnu tačku, bez izmene ostalih ulaznih parametara potrebno je izmeniti samo koeficijent prvog polinoma, dok ostali koeficijenti ostaju neizmenjeni. Naime, na ovaj način neće biti izmenjena početna brzina, krajnja tačka, niti krajnja brzina. Ako bismo, umesto na ovaj način, krivu izrazili kao linearnu kombinaciju baznih vektora standardne baze $\{t^3, t^2, t, 1\}$ izmena rešenja tako da se promeni samo početna tačka rezultovala bi izmenom svih koeficijenata (uz 1, t, t^2 i t^3). Stoga, za ovaj problem, Hermitova baza predstavlja bolji izbor od standardne baze.

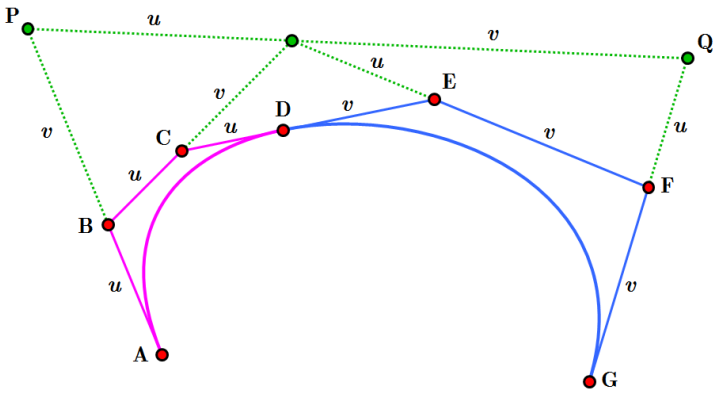
Ako sa $\mathbf{T}(t)$ označimo vektor koji sadrži stepene parametra t, tj. $\mathbf{T}(t) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix}^T$ funkciju $\gamma(t)$ možemo zapisati kao:

$$\gamma(t) = \left[\begin{array}{cccc} P; & Q; & v; & w \end{array}\right] \cdot \begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \cdot \mathbf{T}(t) = G \cdot M \cdot \mathbf{T}(t)$$

Prvi činilac u ovom proizvodu je *geometrijska matrica* krive koju označavamo sa G i ona po kolonama sadrži koordinate tačaka P i Q i vektora v i w. Drugi činilac u ovom proizvodu je tzv. *bazna matrica* koju označavamo sa M i koja po vrstama sadrži koeficijente Hermitovih baznih funkcija, od najnižeg do najvišeg stepena. Ova veza suštinski predstavlja promenu koeficijenata iz Hermitove baze za kubne polinome u koeficijente standardne baze $\{1, t, t^2, t^3\}$.

11.2.2 Bezjeova kriva

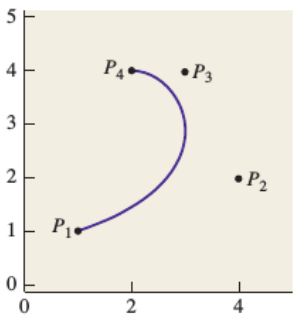
Neka je dat skup kontrolnih tačaka nad kojima želimo da kreiramo glatku krivu. Povezivanjem susednih kontrolnih tačaka dužima dobija se deo po deo linearna aproksimacija koja nije glatka. Dobijeni poligon možemo "popraviti" isecanjem oštrih uglova, čime se dobija novi poligon čiji su uglovi manje oštri ali i dalje ne dovoljno glatka funkcija. Ponavljanjem ovog postupka dobija se sve glatkiji poligon. Ako bismo ovaj postupak ponavljali beskonačno mnogo puta dobili bismo krivu koja je iz klase C^1, odnosno koja je diferencijabilna i čiji je prvi izvod neprekidna funkcija. Na slici 11.3 ilustrovan je ovaj postupak za kontrolne tačke A, P, Q, G: povezivanjem ovih tačaka dobija se kolekcija duži AP, PQ, QG: odsecanjem oštrog ugla kod tačke P i tačke Q dobija se kolekcija duži AB, BC, CE, EF, FG.



Slika 11.3: Postupak "izglađivanja" kubne Bezjeove krive.

Na ovoj ideji počiva *Bezjeova kriva* (eng. Bézier curve) koja je ime dobila po francuskom inženjeru Pjeru Bezjeu (Pierre Bezier), koji ju je koristio šeždesetih godina prošlog veka za dizajniranje karoserije Renoovog automobila.

Bezjeova kriva se zadaje sa četiri tačke P_1, P_2, P_3 i P_4 s tim da kriva počinje u tački P_1, završava se u tački P_4, ima vektor tangente u tački P_1 jednak $3(P_2 - P_1)$, dok je vektor tangente u tački P_4 jednak $3(P_4 - P_3)$ (slika 11.4). Bezjeova kriva prati oblik poligona dobijenog povezivanjem kontrolnih tačaka i sadržana je u konveksnom omotaču skupa kontrolnih tačaka (slika 11.3).



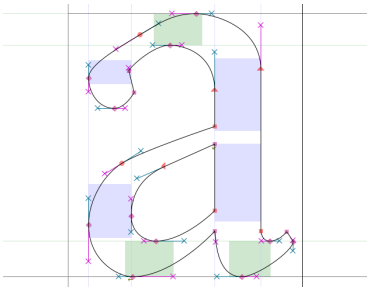
Slika 11.4: Bezjeova kriva počinje u tački P_1, ide ka tački P_2, a završava se u tački P_4 dolazeći iz pravca tačke P_3.

Bezjeova kriva data je jednačinom:

$$\gamma(t) = [P_1; P_2; P_3; P_4] \cdot \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 1 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{T}(t) = G \cdot M \cdot \mathbf{T}(t)$$

pri čemu u ovom slučaju geometrijska matrica G po kolonama sadrži koordinate četiri date tačke, dok bazna matrica M sadrži drugačije koeficijente u odnosu na Hermitovu krivu. Može se učiniti da je specifikacija Bezjeove krive manje prirodna od Hermitove s obzirom na to da je uloga tačaka P_2 i P_3 manje intuitivna u poređenju sa pojmom tangente u početnoj i krajnjoj tački. Međutim, prednost Bezjeove krive je u tome što su svi kontrolni podaci tačke, pa ako želimo da transformišemo Bezjeovu krivu treba samo transformisati tačke.

Bezjeove krive imaju brojne primene. Jedna od važnih primena je za definisanje fontova jer se nadovezivanjem većeg broja Bezjeovih krivih dobijaju putanje koje se mogu lako skalirati, što je u slučaju fontova važna operacija (slika 11.5). Koriste se često i u računarskim igrama za opisivanje putanja poput putanja letova ili trka, za opis karoserije automobila, a takođe i u dizajnu uz pomoć računara (CAD).

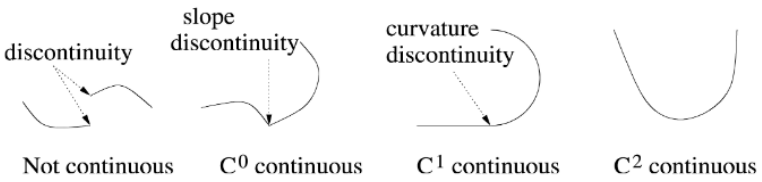


Slika 11.5: Primena Bezjeovih krivih za definisanje fontova.

11.3 Nadovezivanje krivih i splajn

Pretpostavimo da imamo niz tačaka $P_0, P_1, \ldots, P_n$ i pridruženih vektora $v_0, v_1, \ldots, v_n$ i želimo da nađemo krivu $\gamma : [1, n] \to R^2$ koja prolazi kroz tačke P_i sa vektorom brzine v_i (slika 11.7). Ovaj problem se može rešiti povećanjem stepena polinomijalne krive kojom vršimo interpolaciju, međutim, u tom slučaju, kontrola nad krivom nije lokalna: izmena jedne kontrolne tačke menja kompletnu krivu. Alternativno, možemo izvršiti nadovezivanje kubnih krivih u deo po deo kubnu krivu. Na ovaj način postižemo lokalnu kontrolu nad krivom: svaka kontrolna tačka utiče na ograničeni deo krive.

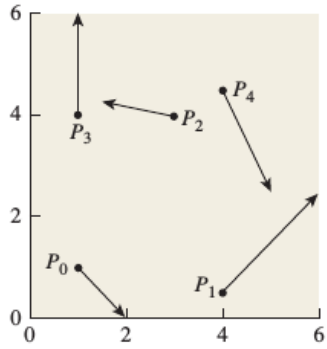
Prilikom nadovezivanja krivih jednu na drugu obično se zahteva neki stepen neprekidnosti u graničnoj (zajedničkoj) tački: može se zahtevati da vrednosti krivih u zajedničkim tačkama budu jednake, ili, dodatno, da tangentni vektori (izvodi) u zajedničkim tačkama budu jednaki ili se može zahtevati i više: da i prvi i drugi izvod u tačkama spoja budu jednaki.



Slika 11.6: Primer (a) krive koja nije neprekidna, (b) krive iz klase C^0 (c) krive iz klase C^1 (d) krive iz klase C^2.

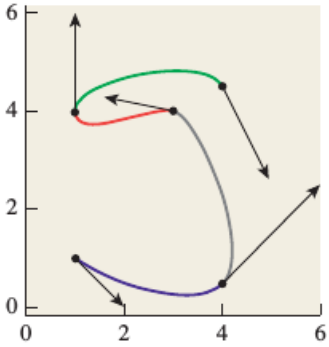
Podsetimo se da kriva pripada klasi C^0 ako je neprekidna. Ako je kriva diferencijabilna i njen prvi izvod je neprekidan onda ona pripada klasi C^1, a ako je dvaput diferencijabilna i njeni prvi i drugi izvod su neprekidni onda pripada klasi C^2 (slika 11.6). Na sličan način se definišu i klase C^n za $n > 2$.

Dakle, ukoliko se dva segmenta krive nadovežu jedan na drugi, rezultujuća kriva biće iz klase C^0. Ako su vektori tangenti segmenata krive jednaki u spojnoj tački, onda rezultujuća kriva pripada klasi C^1.



Slika 11.7: Za dati niz tačaka $P_0, P_1, \ldots, P_4$ i odgovarajućih vektora treba odrediti krivu koja prolazi kroz tačku P_i sa datim vektorima kao vektorima brzine.

Vratimo se polaznom problemu: da konstruišemo krivu koja prolazi kroz date tačke P_i sa datim vektorima brzine v_i. Za njegovo rešavanje možemo iskoristiti Hermitovu formulaciju. Naime, možemo da odredimo krivu $\gamma_0 : [0,1] \to R^2$ koja počinje u tački P_0, završava se u tački P_1 i ima vektore tangenti u početnoj i završnoj tački redom jednake v_0 i v_1. Možemo, takođe, odrediti krivu $\gamma_1 : [0,1] \to R^2$ koja počinje u tački P_1, završava se u tački P_2 i ima odgovarajuće vektore tangenti v_1 i v_2 i slično pronaći krive $\gamma_2, \ldots, \gamma_{n-1}$. Nadovezivanjem krivih $\gamma_0, \gamma_1, \ldots, \gamma_{n-1}$ dobija se kriva γ koja je neprekidna i diferencijabilna i koja prolazi kroz svaku od zadatih tačaka sa odgovarajućim vektorom tangente (slika 11.8). Pojedinačni delovi kolekcije $t_i \to \gamma_i(t-i)$ nazivaju se *segmentima krive* (eng. segments), kompletna kolekcija *splajn* (eng. spline), a date tačke i vektori *kontrolnim podacima* (eng. control data). Dakle, splajn je parametarska kriva vođena kontrolnim tačkama ili kontrolnim vektorima, reda tri ili više.



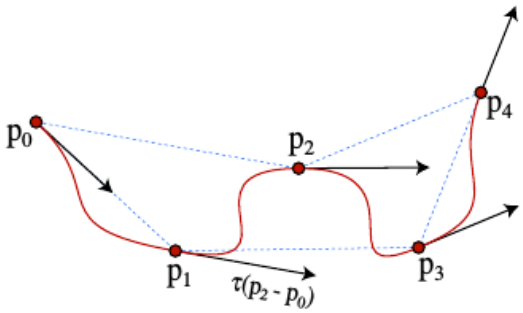
Slika 11.8: Kolekcija segmenata krive koja formira splajn kao rešenje problema.

11.3.1 Katmul-Rom splajn

Razmotrimo naredni problem: za dati niz tačaka $P_0, P_1, \ldots, P_n$ potrebno je pronaći glatku krivu koja prolazi kroz tačku P_i u trenutku $t = i$, sa svojstvom da ako su sve tačke na istoj pravoj na istom rastojanju rezultujuća kriva je samo prava linija interpolacije od prve do poslednje tačke.

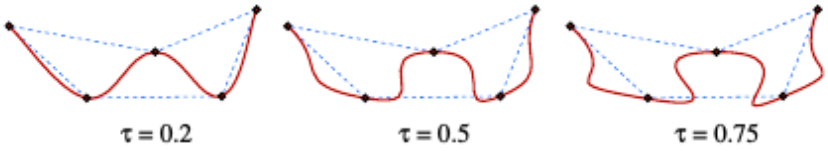
S obzirom na to da su u ovoj formulaciji jedini zadati kontrolni podaci tačke, možemo pomisliti da je kroz svake četiri susedne tačke moguće provući kubnu Bezjeovu krivu, međutim, potrebno je da kriva prođe kroz sve zadate tačke, a znamo da kubna Bezjeova kriva prolazi samo kroz dve od datih četiri tačaka. Ako bismo u svakoj od tačaka P_i imali na raspolaganju vektor tangente mogli bismo da koristimo Hermitovu krivu za konstrukciju svakog pojedinačnog segmenta krive. S obzirom na to da vektori tangenti nisu zadati, jedno moguće rešenje je da se za navođenje u tekućoj tački koristi prethodna i naredna kontrolna tačka. Preciznije, tangentni vektor u tački P_i se bira tako da bude kolinearan vektoru $P_{i+1} - P_{i-1}$ (slika 11.9). Ova vrsta krive naziva se *Katmul-Rom splajn* i njega su 1974. godine formulisali Edvin Katmul (Edwin Catmull) i Rafael Rom (Raphael Rom).

Vektor tangente $P_{i+1} - P_{i-1}$ se najčešće skalira nekom vrednošću $\tau \in [0,1]$ koju nazivamo *tenzijom splajna* (eng. tension). Tenzija splajna određuje koliko oštro se kriva savija u kontrolnim tačkama (slika 11.10): što je tenzija splajna manja, to se kriva oštrije savija u kontrolnim tačkama. Često se kao tenzija



Slika 11.9: Primer Katmul-Rom splajna.

splajna uzima vrednost $\tau = 1/2$, odnosno tangentni vektor u tački P_i imaće vrednost $v_i = \frac{1}{2}(P_{i+1} - P_{i-1})$.



Slika 11.10: Efekat odabira različitih vrednosti za tenziju splajna.

Dakle, krivu na segmentu P_iP_{i+1} možemo odrediti tako što zamenimo gornje vrednosti vektora tangenti u jednačinu Hermitove krive. Na taj način dobijamo narednu jednačinu odgovarajućeg segmenta krive:

$$\gamma_i(t) = \left[\begin{array}{cccc} P_i; & P_{i+1}; & \frac{P_{i+1}-P_{i-1}}{2}; & \frac{P_{i+2}-P_i}{2} \end{array} \right] \cdot \begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \cdot \mathbf{T}(t)$$

odakle dobijamo vrednost $\gamma_i(t)$ kao funkciju tačaka P_{i-1}, P_i, P_{i+1} i P_{i+2}.

Pošto za prvu tačku ne postoji njoj prethodna niti za poslednju njoj naredna tačka, za tačku P_0 koristimo vektor tangente $v_0 = \frac{1}{2}(P_1 - P_0)$ (koji bismo dobili kada bi postojala i tačka P_{-1} simetrično tački P_1 u odnosu na tačku P_0). Slično dobijamo i vektor tangente $v_n = \frac{1}{2}(P_n - P_{n-1})$ za tačku P_n.

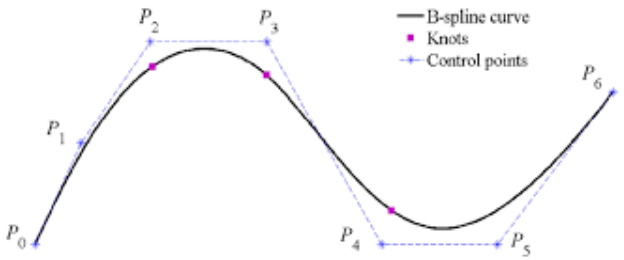
Pojedinačne funkcije u splajnu su beskonačno diferencijabilne u većini tačaka (jer su definisane polinomima), ali su u spojnim tačkama samo jednom diferencijabilne. Ovo je prihvatljivo za neke primene, ali recimo ne bi bilo pogodno za definisanje putanje kamere koja treba da prođe kroz unapred definisan skup kontrolnih tačaka jer bi rezultujuće kretanje posmatraču delovalo "skokovito". Takođe, ova vrsta splajna ne leži unutar konveksnog omotača kontrolnih tačaka.

11.3.2 Kubni B-splajn

Pored razmatranog, postoje i drugi tipovi splajnova. U ovom poglavlju razmotrićemo *kubni B-splajn*[1].

Kubni B-splajn zadaje se nizom od $m + 1$ kontrolnih tačaka $P_0, P_1, \ldots, P_m$ za $m \geq 3$ i sastoji se od $m-2$ segmenata krive koji su predstavljeni polinomima trećeg stepena. Svaki segment se može definisati na svom domenu $0 \leq t \leq 1$. i-ti segment ove krive za $3 \leq i \leq m$ definisan je tačkama $P_{i-3}, P_{i-2}, P_{i-1}$ i P_i. Ova vrsta splajna ne prolazi nužno kroz same kontrolne tačke (dobijeni splajn je neinterpolišući). U slučaju kada je $m \geq 4$, za svako $3 < i \leq m$ zadaju se tačke u kojima se spajaju dva segmenta krive i koje se nazivaju *čvorovima* (slika 11.11). U čvorove B-splajna spadaju i početna i krajnja tačka splajna, te splajn ima ukupno $m - 1$ čvorova. Svaki od segmenata B-splajna zadat je korišćenjem četiri od ukupno $m + 1$ kontrolne tačke. Svaka od kontrolnih tačaka utiče na najviše četiri segmenta krive, te se izmenom jedne kontrolne tačke menjaju najviše četiri segmenta krive, dok na ostale segmente ova izmena neće imati uticaja. Kubni B-splajn je za razliku od Katmul-Rom splajna C^2 gladak, tj. i prvi i drugi izvod su neprekidne funkcije. Kao i Katmul-Rom splajn, on omogućava lokalnu kontrolu krive, odnosno promena jedne kontrolne tačke utiče na samo mali deo krive.

[1] Pored kubnih, postoje i linearni, kvadratni i drugi B-splajnovi, ali se kubni najviše koriste.



Slika 11.11: Kubni B-splajn definisan nad skupom kontrolnih tačaka $P_0, P_1, \ldots, P_6$ koji se sastoji od četiri segmenta koji se spajaju u datim čvorovima.

U zavisnosti od toga da li su čvorovi smešteni na istim intervalima u odnosu na parametar t razlikuju se uniformni i neuniformni B-splajnovi.

Iako B-splajnovi ne prolaze kroz kontrolne tačke, njihov dodatni stepen neprekidnosti čini ih atraktivnim za mnoge primene.

11.4 Crtanje krivih

Postoje dva osnovna pristupa crtanju krivih. U prvom pristupu vrednost parametra t se povećava za konstantnu (malu) vrednost i tačka $(x(t), y(t), z(t))$ se pravom linijom spaja sa prethodno određenom tačkom. Ovaj pristup se može efikasno implementirati imajući u vidu da je kriva opisana polinomom trećeg stepena. Umesto računanja vrednosti polinoma za svaku novu vrednost parametra t Hornerovom shemom, moguće je koristiti pristup preko razlika unapred, razlika drugog i razlika trećeg reda (kojima se eliminiše potreba za operacijom množenja).

Drugi pristup vrši rekurzivnu podelu krive na segmente sve dok se ne dođe do segmenta koji se može aproksimirati pravom linijom. Ova podela se različito implementira za različite tipove krivih, a razlikuju se i testovi da li se segment može aproksimirati pravom linijom.

11.5 Opisivanje površi u trodimenzionom prostoru

Parametarska reprezentacija površi u trodimenzionom prostoru predstavlja uopštenje parametarske reprezentacije krivih.

Podsetimo se da parametarsku krivu trećeg stepena možemo predstaviti matrično u sledećem obliku

$$\gamma(s) = G \cdot M \cdot \mathbf{S}(s)$$

gde je sa s označena vrednost parametra, $\mathbf{S}(s) = \begin{bmatrix} 1 & s & s^2 & s^3 \end{bmatrix}^T$, G je geometrijska matrica krive, a M bazna matrica. Umesto uobičajene oznake t parametra, koristimo oznaku s kako bismo znali da se radi o površima.

U slučaju opisivanja površi, umesto konstantne vrednosti geometrijske matrice, dozvoljavamo da se tačke u geometrijskoj matrici menjaju u trodimenzionom prostoru duž neke putanje, čime dobijamo naredni izraz:

$$\gamma(s, t) = G(t) \cdot M \cdot S(s)$$

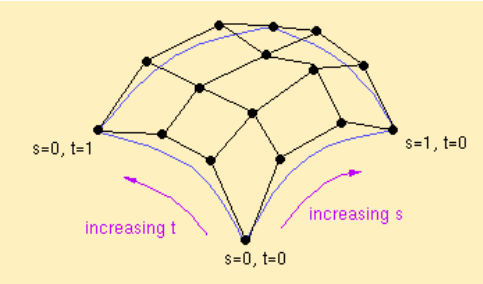
Za fiksiranu vrednost t_1 parametra t, $\gamma(s, t_1)$ predstavljaće parametarsku krivu. Takođe, za vrednost t_2 blisku vrednosti t_1 dobija se kriva $\gamma(s, t_2)$ bliska krivoj $\gamma(s, t_1)$. Ako ovaj postupak ponovimo za različite vrednosti parametra t iz opsega $[0, 1]$ dobiće se familija krivih koja formira površ u trodimenzionom prostoru (slika 11.12). Ako su u pitanju krive trećeg stepena, dobija se *parametarska površ trećeg stepena*.

Za različite izbore geometrijske matrice, dobijaju se različite vrste površi kao što su Hermitova površ, Bezjeova površ i B-splajn površ.

11.6 Pitanja

11.1 Da li je pogodno čuvati krive u rasterskom obliku?

11.2 Zašto deo po deo linearna funkcija ni trigonometrijske funkcije ne predstavljaju dobar izbor za zadavanje krive? Zašto aproksimacija polinomima stepena 3 predstavlja pogodan izbor?



Slika 11.12: Parametarski zadata površ u trodimenzionom prostoru.

11.3 Zašto je parametarska reprezentacija krive bolja od eksplicitne i implicitne reprezentacije?

11.4 Definisati Hermitovu krivu.

11.5 Šta sadrži geometrijska matrica Hermitove krive i čime je određena?

11.6 Definisati Bezjeovu krivu.

11.7 Bezjeova kriva zadata je: (a) kontrolnim tačkama (b) kontrolnim tačkama i kontrolnim vektorima.

11.8 Bezjeova kriva prolazi kroz (a) 0 (b) 1 (c) 2 (d) 3 (e) 4 date kontrolne tačke.

11.9 Šta je splajn?

11.10 Šta je tenzija splajna?

11.11 Da li je kod Katmul-Rom splajna dat vektor tangente u svakoj od kontrolnih tačaka? Čemu je jednak vektor tangente u svakoj od datih tačaka?

11.12 Da li je kriva koja je klase C^1 nužno i klase (a) C^0 (da/ne) (b) C^2 (da/ne)?

11.13 Katmul-Rom splajn pripada klasi (a) C^0 (b) C^1 (c) C^2.

11.14 Navesti razlike između Katmul-Rom splajna i B-splajna.

11.15 Koja su dva moguća načina za crtanje krivih?

11.16 Na koji način možemo parametarski zadati površ u trodimenzionom prostoru? Sa koliko parametara je ona parametrizovana?

Glava 12

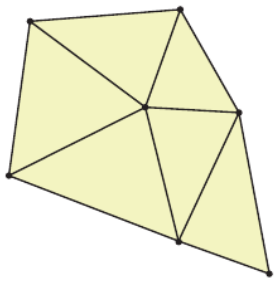
Reprezentacija figura

U ovom poglavlju bavićemo se opisivanjem figura u ravni i trodimenzionom prostoru.

12.1 Mreže trouglova

Mreže poligona (eng. polygon mesh) predstavljaju jednu od najkorišćenijih reprezentacija figura u računarskoj grafici. Mreža poligona se sastoji od konačnog, ali često jako velikog broja poligona međusobno povezanih ivicama koji na taj način formiraju površ. U ovoj vrsti reprezentacije za svako teme poznati su svi njegovi susedi. Skoro sve figure moguće je egzaktno ili približno (ako je površ zakrivljena) predstaviti mrežom poligona. Činjenica da su strane mreže poligona planarne omogućava brzo renderovanje primitiva jer paralelne prave zahvataju isti ugao sa svim tačkama poligona. Konačno, veliki broj računarskih alata na ulazu prima ili kao izlaz daje ovakvu reprezentaciju figure.

Najčešće se koriste *mreže trouglova* (eng. triangle mesh) (slika 12.1). Jedan od razloga za to je taj što je geometrija trougla veoma dobro izučena.



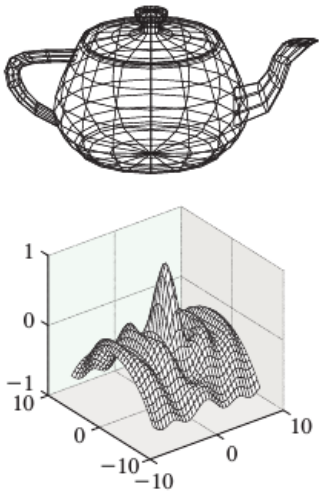
Slika 12.1: Mreža trouglova.

Umesto trouglova, ponekad se koriste i mreže četvorouglova ili drugih poligona (slika 12.2), međutim, pri njihovoj upotrebi mogu se uočiti neki problemi. Naime, četiri tačke u trodimenzionom prostoru ne moraju biti komplanarne, dok se ovo ne može desiti u slučaju trougla jer trougao uvek leži u ravni.

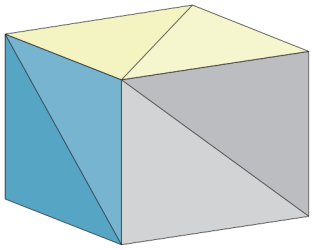
Određene figure moguće je egzaktno opisati korišćenjem mreža trouglova. Naime, svaki poligon može se izdeliti na trouglove, te se svaki poliedar može predstaviti mrežom poligona podelom njegovih strana na trouglove. Na slici 12.3 prikazan je model kocke predstavljen u vidu mreže trouglova. Neke figure pak nije moguće precizno predstaviti mrežom trouglova, ali ih je moguće dovoljno dobro aproksimirati. Jedan način da se ovo postigne jeste da se utvrde lokacije velikog broja tačaka na figuri, a da se zatim susedne lokacije povežu mrežnom strukturom. Ovakva aproksimacija, ukoliko su tačke dovoljno blizu jedna drugoj, može dati privid glatke površi (slika 12.2). Međutim nisu svi objekti pogodni za predstavljanje mrežama trouglova. Za neke oblike je, na primer, karakteristično da imaju visok nivo detaljnosti u svakoj razmeri (recimo šljunak) i kao takve nije ih moguće jednostavno predstaviti mrežom trouglova.

Mreža trouglova ne može sadržati ivicu koja "visi", odnosno ivicu koja nije stranica nijednog trougla (slika 12.4).

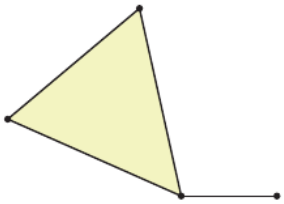
Trouglovi idealne mreže trouglova su bliski jednakostraničnim i broj suseda svakog od temena mreže je približno jednak 6 (slika 12.5).



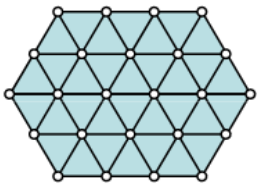
Slika 12.2: Korišćenje mreža poligona za predstavljanje čajnika i glatke talasaste površi.



Slika 12.3: Mreža trouglova koja ima geometriju kocke.



Slika 12.4: Trougao sa dodatnom ivicom koja "visi" ne može biti deo strukture mreže.

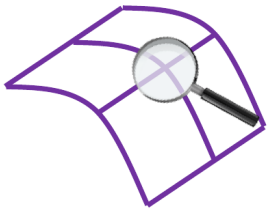


Slika 12.5: Fragment idealne mreže trouglova.

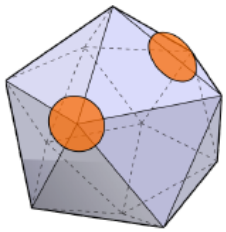
Za dvodimenzionu mrežu kažemo da je *mreža mnogostrukosti* ako za svako teme v mreže važi da se ivice i trouglovi ove mreže susedni temenu v mogu poređati u ciklični poredak: $t_1, e_1, t_2, e_2, \ldots, t_n, e_n$ bez ponavljanja tako da je ivica e_i ivica trouglova t_i i t_{i+1}. Dakle, za svaku ivicu mreže mnogostrukosti postoje tačno dve strane mreže kojima ona pripada. Ovo je u skladu sa pojmom dvodimenzione mnogostrukosti u topologiji koja se definiše tako da svaka tačka ima okolinu koja izgleda kao disk. Drugim rečima, ako površ dovoljno zumiramo u proizvoljnoj tački, delovaće kao ravan (slika 12.6). Primer ovoga je Zemljina kugla. Ako mrežu mnogostrukosti u proizvoljnoj tački presečemo malom sferom dobijamo disk.

Na slici 12.7 prikazana je mreža trouglova koja je uz to i mreža mnogostrukosti. Primetimo da okolina svake tačke izgleda kao disk, tj. ako je presečemo dovoljno malom sferom presek nalikuje disku.

Na slici 12.8 prikazane su dve mreže trouglova koje nisu mreže mnogostrukosti. Naime, za prvu od

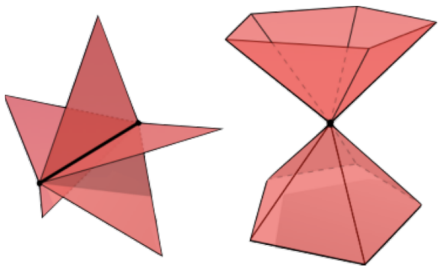


Slika 12.6: Zumiranjem mreže mnogostrukosti u proizvoljnoj tački dobijamo privid ravni.



Slika 12.7: Primer mreže mnogostrukosti.

njih važi da postoji ivica koja pripada pet trouglova; što se druge mreže tiče postoji teme čije se susedne ivice i trouglovi ne mogu poređati u jedan ciklični niz, već dva.



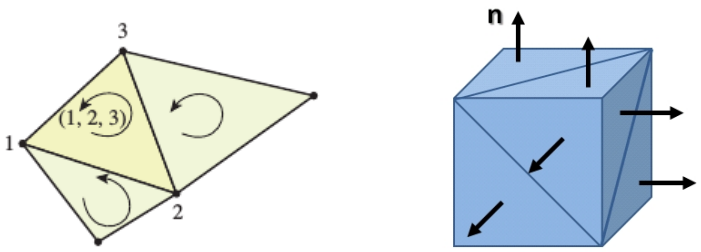
Slika 12.8: Primeri mreža koje nisu mreže mnogostrukosti.

Mreže mnogostrukosti ne dozvoljavaju dodavanja i brisanja trouglova jer bi svako dodavanje ili brisanje trougla iz mreže pokvarilo svojstva ove mreže.

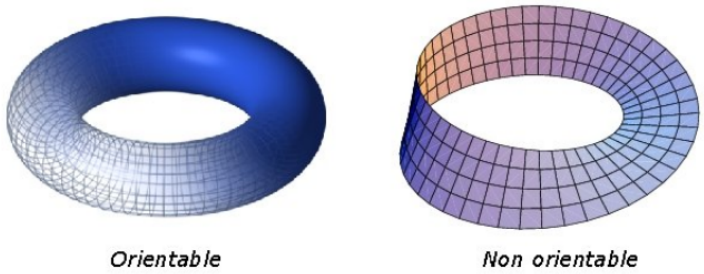
Često će nam biti od značaja da vodimo računa o *orijentaciji* trouglova u mreži. Vektori normale trougla su važni za razne primene, npr. prilikom određivanja zadnje strane objekta, za računanje interakcije svetlosti i površi i sl. Orijentaciju trougla u mreži možemo zadati njegovim vektorom normale: naime, ako su temena datog trougla tačke P_i, P_j i P_k onda je potrebno izračunati vektor $(\vec{P_iP_j} \times \vec{P_iP_k}$ koji je upravan na ravan kojoj pripada trougao. Kažemo da dva susedna trougla mreže $\triangle P_iP_jP_k$ i $\triangle P_lP_jP_i$ imaju *konzistentno orijentisane* vektore normale ako se ivica koju dele javlja kao (P_i, P_j) u jednom trouglu, a kao (P_j, P_i) u drugom. Drugim rečima vektori normale ova dva trougla biće usmereni na istu stranu mreže. Trouglovi sa slike 12.9 imaju konzistentno orijentisane vektore normala.

Primetimo i to da postoje mreže poligona kod kojih nije moguće konzistentno orijentisati vektore normala susednih poligona: za takve mreže poligona kažemo da su *neorijentabilne* (slika 12.10).

Mreže mnogostrukosti su pogodne za rad, ali su za neke primene previše krute (npr. ne dozvoljavaju umetanje i brisanje trouglova) i u nekim situacijama poželjno je razmatrati mreže kod kojih su uslovi nešto relaksiraniji. Jedna mogućnost za to je dozvoliti da mreže imaju granicu. Ovakve mreže nisu mreže mnogostrukosti jer tačka na granici ima okolinu koja je sa jedne strane "odsečena". Međutim, jednostavnosti radi mi ćemo ih u daljem tekstu nazivati *mrežama mnogostruktosti sa granicom*. Dakle, to su mreže kod kojih umesto da trouglovi susedni nekom temenu obrazuju ciklus, oni obrazuju lanac čiji prvi i poslednji element dele samo jednu svoju ivicu sa ostalim trouglovima u lancu. Druga ivica prvog trougla u lancu sadržana je u prvom trouglu i ni u jednom drugom trouglu mreže (slično važi i za poslednji trougao u lancu (slika 12.11). Ova nedeljena ivica se naziva *graničnom ivicom* a teme čiji

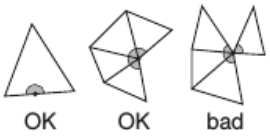


Slika 12.9: Dve mreže koje imaju konzistentno orijentisane vektore normala. Primetimo da se u prvoj mreži ivica $(2, 3)$ javlja kao ivica jednog trougla, a ivica $(3, 2)$ kao ivica drugog trougla.



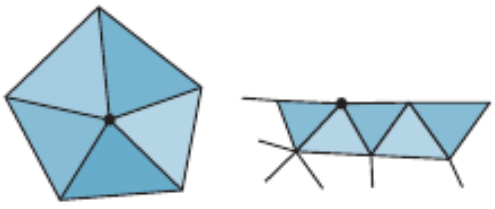
Slika 12.10: Razlika između orijentabilne i neorijentabilne mreže poligona.

susedni trouglovi obrazuju lanac *graničnim temenom*. Temena koja nisu granična nazivamo *unutrašnjim temenima*. Za mreže mnogostrukosti sa granicom važe sledeći uslovi:



Slika 12.11: Primeri mreže mnogostrukosti sa granicom (prve dve slike) i mreže koja to nije (treća slika).

- svaka ivica je sadržana u jednom ili dva trougla,
- svako teme je sadržano u jednom lancu trouglova povezanih ivicama.



Slika 12.12: Teme mreže mnogostrukosti i granično teme mreže mnogostrukosti sa granicom.

U slučaju kad je mreža orijentisana, granicu trougla mreže sa temenima v_i, v_j i v_k možemo definisati kao sumu granica ivica

$$(v_i, v_j) + (v_j, v_k) + (v_k, v_i)$$

. Granicu kolekcije orijentisanih trouglova možemo definisati kao sumu granica pojedinačnih trouglova. Granica orijentisane mreže mnogostrukosti ima vrednost 0 jer ako je ivica (v_i, v_j) deo granice jedne strane mreže, ivica (v_j, v_i) biće deo granice druge strane. Za orijentisane mreže mnogostrukosti

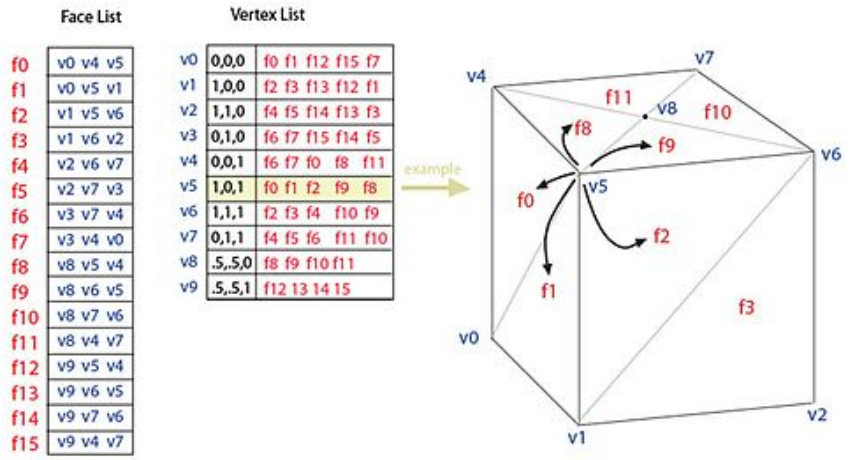
sa granicom, granica će se sastojati od tačno onih ivica koje smo identifikovali kao granične ivice. U opštem slučaju, orijentisanu mrežu mnogostrukosti bez graničnih ivica nazivamo *zatvorenom*.

U nastavku teksta razmotrićemo različite reprezentacije mreža trouglova i videti kakvi su njihovi memorijski zahtevi i koje su složenosti karakterističnih operacija za svaku od tih reprezentacija. Takođe ćemo razmotrićemo na koji način možemo popraviti svojstva postojeće mreže trouglova.

12.2 Različite reprezentacije mreže poligona

Razmotrićemo više različitih struktura podataka kojima je moguće predstaviti mreže poligona. One se međusobno razlikuju prema memorijskim zahtevima potrebnim za čuvanje elemenata mreže, vremenu potrebnom za renderovanje mreže, efikasnosti izvršavanja karakterističnih upita (npr. koja su temena susedna datoj strani mreže i koje su strane susedne datoj strani mreže) i operacija izmene mreže (dodavanja ili brisanja elemenata mreže).

Dvodimenzionu mrežu možemo čuvati tabelom temena i tabelom trouglova. Tabela temena čuva za svako teme njegove koordinate, a tabela trouglova za svaki trougao listu indeksa temena tog trougla (slika 12.13). Dodavanje novih temena i novih trouglova u mreu se izvršava efikasno, dok je brisanje temena iz mreže (koje povlači i brisanje svih trouglova susednih tom temenu) neefikasna operacija. Možemo održavati i tabelu susedstva koja za svako teme čuva trouglove koji sadrže to teme i na taj način operaciju brisanja temena mreže učiniti efikasnijom. U opštem slučaju ta lista je neuređena, međutim, ako nametnemo neke dodatne uslove na mrežu onda lista susednih trouglova datom temenu može biti i uređena.



Slika 12.13: Primer tabele temena, tabele trouglova i tabele susedstva (koja je na slici integrisana sa tabelom trouglova).

U ovoj strukturi podataka dodavanje novih temena i trouglova se efikasno izvršava, ali je brisanje temena sporo (jer je potrebno naći sve pridružene trouglove i obrisati ih).

Zbog toga što se ivice zasebno ne pamte, u dvodimenzionu mrežu nije moguće dodati novu ivicu. Jedino možemo ispitivati da li je neka duž ivica mreže i ovo ispitivanje bilo bi složenosti $O(T)$, gde je T broj trouglova, jer je potrebno uraditi iscrpnu pretragu svih trouglova u mreži. Ipak, u nekim specijalnim slučajevima, ove operacije se mogu učiniti efikasnijim.

12.2.1 Eksplicitna reprezentacija

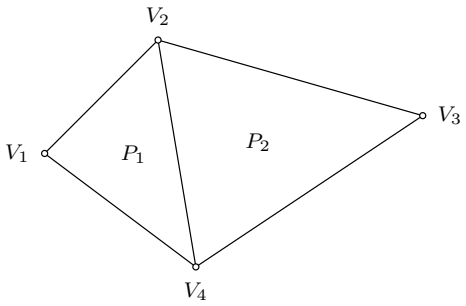
U *eksplicitnoj reprezentaciji* koja se još naziva i *supa poligona*, svaki poligon je opisan nizom koordinata svojih temena. Za jedan poligon ovo rešenje je prostorno efikasno, međutim, za mreže koje sadrže veliki broj poligona ovo rešenje može da troši mnogo prostora jer veliki broj temena može da pripada većem broju poligona iz skupa.

Veliki problem u ovoj reprezentaciji predstavlja i to što ne postoji eksplicitna informacija o zajedničkim temenima i ivicama. Da bi se proverilo da li je jedno teme istovremeno i teme nekog drugog poligona treba proveriti sva njegova temena. Štaviše, ta provera može da bude nepouzdana jer zbog grešaka u računu dve reprezentacije (u dva poligona) jedne iste tačke mogu da se razlikuju.

Prilikom iscrtavanja mreže poligona u ovoj reprezentaciji svaka ivica biće (nepotrebno) iscrtavana dva puta.

12.2.2 Reprezentacija sa pokazivačima na liste indeksa temena

U *reprezentaciji sa pokazivačima na liste indeksa temena* svako teme je zapisano tačno jednom u listi temena, dok je poligon predstavljen listom indeksa temena (slika 12.14). Ova reprezentacija je memorijski efikasnija od prethodne. Dodatno, ona omogućava ispitivanje da li dva poligona imaju zajedničko teme. Međutim, i u ovoj reprezentaciji ne postoji eksplicitna inforamcija o zajedničkim temenima i ivicama. Kao i u prethodnoj, i u ovoj reprezentaciji se pri iscrtavanju mreže poligona svaka ivica crta dva puta.



$V = (V_1, V_2, V_3, V_4) = ((x_1, y_1, z_1), ..$

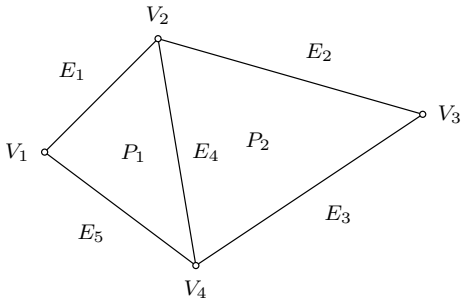
$P_1 = (1, 2, 4)$

$P_2 = (4, 2, 3)$

Slika 12.14: Reprezentacija sa pokazivačima na liste indeksa temena.

12.2.3 Reprezentacija sa pokazivačima na liste indeksa ivica

Mrežu poligona moguće je predstaviti i korišćenjem *reprezentacije sa pokazivačima na liste indeksa ivica*. U ovoj reprezentaciji poligoni su predstavljeni kao liste indeksa ivica, dok je svaka ivica opisana parom svojih temena (slika 12.15). Dodatno, u opisu ivice mogu biti zadati i poligoni kojima ona pripada (radi efikasnijih obrada). Za razliku od prethodne dve reprezentacije, u ovoj reprezentaciji se pri iscrtavanju mreže svaka ivica iscrtava tačno jednom.



$V = (V_1, V_2, V_3, V_4) = ((x_1, y_1, z_1), ..$

$E_1 = (V_1, V_2, P_1, null)$

$E_2 = (V_2, V_3, P_2, null)$

$E_3 = (V_3, V_4, P_2, null)$

$E_4 = (V_4, V_2, P_1, P2)$

$E_5 = (V_4, V_1, P_1, null)$

$P_1 = (E_1, E_4, E_5)$

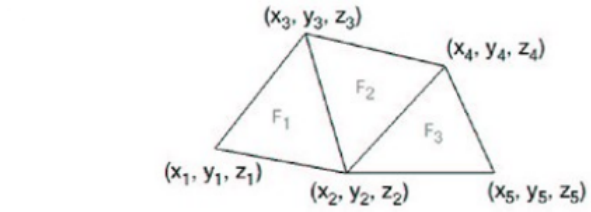
$P_2 = (E_2, E_3, E_4)$

Slika 12.15: Reprezentacija sa pokazivačima na liste indeksa ivica.

12.2.4 Winged-edge reprezentacija

U slučajevima mreža kod kojih strane ne moraju biti trouglovi, može se koristiti i *winged-edge* struktura podataka. Ova struktura podataka posmatra ivice kao "građane prvog reda", odnosno gotovo sve informacije se pamte putem ivica mreže. Naime, za svaku ivicu čuvaju se indeksi dva temena koje ta ivica povezuje, na dve strane (levu i desnu) čiji je ona deo i na prethodnu i sledeću ivicu prilikom obilaska u smeru obrnutom od smera kazaljke na časovniku njegove leve i desne strane (slika 12.16). Svako teme i strana, takođe, sadrže indeks jedne, proizvoljne ivice mreže incidentne sa datim temenom, odnosno stranom mreže.

Iako se nigde eksplicitno ne pamti lista temena nekog poligona, u ovoj reprezentaciji se za datu stranu mreže mogu efikasno odrediti sva njena temena i za dato teme odrediti sve susedne ivice ili strane.



VERTEX TABLE

V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

EDGE TABLE

					11	12	21	22
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	V_3	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

FACE TABLE

F_1	e_1
F_2	e_3
F_3	e_5

Slika 12.16: Winged-edge struktura podataka.

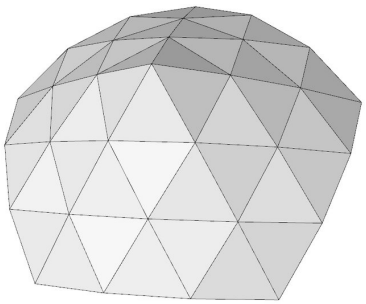
Primer 12.1. Razmotrimo kako za teme v odrediti listu ivica $(e_1, e_2, \ldots, e_r)$ incidentnih sa tim temenom poređanih u smeru suprotnom od kretanja kazaljke na časovniku. Krećemo od ivice e_1 koja je zadata kao susedna temenu v. Za ivicu e_1 možemo učitati četiri ivice $(e_2', e_2'', e_3', e_3'')$ koje prethode i slede nakon ivice e_1 u obilasku strana susednih ivici e_1. Takođe, za ivicu e_1 možemo pročitati njena temena v_0 i v_1. U zavisnosti od toga da li je $v = v_0$ ili $v = v_1$, kao narednu ivicu postavljamo ivicu e_2' ili e_2'' i nastavljamo na isti način sve dok se ne vratimo do ivice e_1.

Svaka od navedenih reprezentacija ima prednosti i mane i izbor reprezentacije treba napraviti u skladu sa konkretnim zadatkom.

12.2.5 Struktura podataka zasnovana na poluivicama

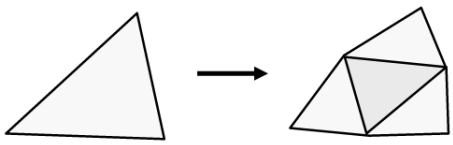
12.3 Operacije nad mrežom trouglova

U ovom poglavlju razmatraćemo kako datu mrežu trouglova popraviti tako da predstavlja bolju diskretnu reprezentaciju odgovarajuće površi. Idealna mreža trouglova je ona čije su sve ivice jednake dužine, čije su sve strane u obliku jednakostraničnih trouglova, a broj suseda svakog temena je blizak broju 6 (slika 12.17). U nastavku teksta razmotrićemo različite operacije nad mrežom trouglova kojima je moguće popraviti svojstva trenutne mreže.



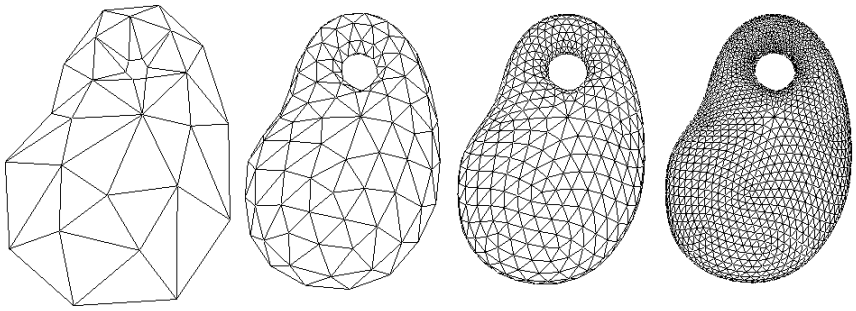
Slika 12.17: Mreža jednakostraničnih trouglova kod koje svako teme ima 5 ili 6 suseda.

Jedna od najznačajnijih osobina mreže trouglova jeste njena uniformnost. Ona nam omogućava primenu različitih operacija sa garancijama koje je lako dokazati. Razlikujemo dve osnovne vrste operacija koje se mogu izvršiti nad mrežom trouglova: to su *unapređivanje mreže* (eng. mesh refinement) i *pojednostavljivanje mreže* (eng. mesh simplification). Unapređivanjem mreže generiše se mreža sa većim brojem temena, ivica ili strana, dok se pojednostavljivanjem proizvodi mreža sa manjim brojem temena, ivica ili strana. Pored toga postoje i operacije kojima se unapređuje struktura mreže, ali se pritom ne menja broj elemenata mreže.



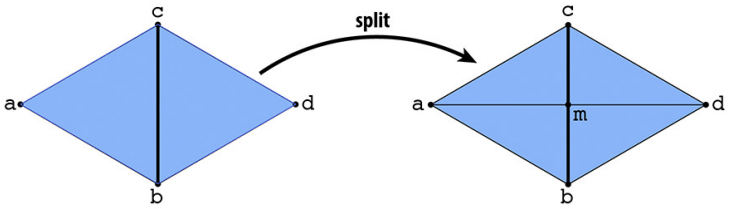
Slika 12.18: Operacija podele trougla.

Jedna od operacija kojom se mreža unapređuje je *podela trougla* (eng. triangle split) kojom se jedan trougao zamenjuje sa nekoliko manjih trouglova (slika 12.18). Podela se obično koristi kako bi se mreža koja ima oštre uglove učinila glatkijom. Naravno, ponovljenim podelama značajno se povećava broj trouglova u mreži što može značajno uticati na vreme renderovanja mreže (slika 12.19).

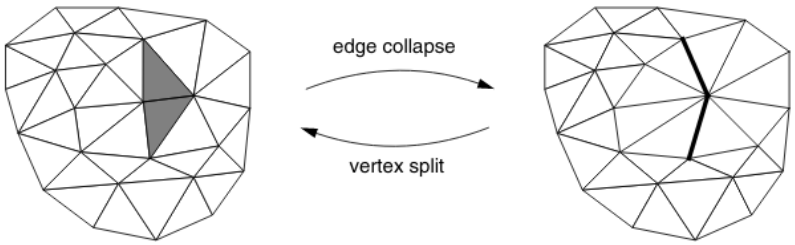


Slika 12.19: Ilustracija višestruke primene operacije podele trougla.

Pored podele trougla, moguće je vršiti i *podelu ivice* (eng. edge split) kojom se dodaje novo teme v na ivici e i teme v se povezuje sa naspramnim temenima trouglova koji su incidentni sa ivicom e (slika 12.20). Na taj način se svaki od ovih trouglova deli na dva. Moguće je vršiti i *podelu temena* (eng. vertex split) kojom se dato teme duplira, kao i dve ivice susedne tom temenu (slika 12.21).



Slika 12.20: Operacija podele ivice.

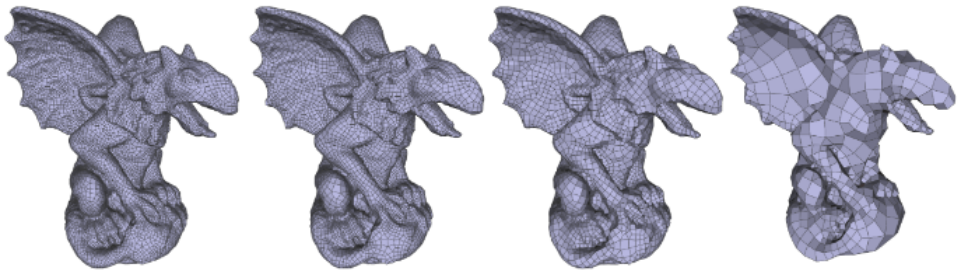


Slika 12.21: Operacija podele temena i njoj inverzna operacija sažimanja ivice.

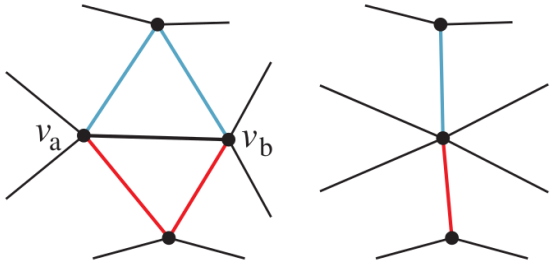
Operacijama pojednostavljivanja mreža se zamenjuje drugom mrežom koja joj je slična (topološki ili geometrijski) ali koja ima kompaktniju strukturu (slika 12.22). Uzastopnim izvršavanjem ove operacije možemo doći do sve jednostavnije reprezentacije iste površi.

Jedna od standardnih operacija prilikom pojednostavljivanja mreže je *sažimanje ivice* (eng. edge collapse), kojom se ivica sažima sve dok njena dužina ne postane 0 čime dva susedna trougla nestaju (slika 12.23). Ova operacija je inverzna operaciji podele temena.

Prilikom stapanja dva temena u jedno potrebno je izabrati lokaciju za novo teme. Lokacija zavisi od željenog cilja: ako želimo da minimizujemo račun, onda možemo za lokaciju novog temena izabrati



Slika 12.22: Pojednostavljivanje mreže trouglova.

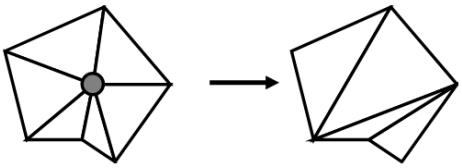


Slika 12.23: Sažimanje ivice: ivica $v_a v_b$ se sažima sve dok joj dužina ne postane 0.

jedno od dva stara temena; ukoliko želimo da očuvamo oblik, za novo teme možemo da izaberemo središte duži određene ovim temenima. Međutim, ako ovakvo uprosečavanje pomera veliki broj tačaka a to nije vizuelno poželjno, lokaciju nove tačke možemo da izaberemo tako da se minimizuje maksimalno rastojanje tačaka mreže od nove tačke mreže (slika 12.24).

Slika 12.24: Različiti geometrijski izbori za lokaciju novog temena prilikom sažimanja ivice u ravni.

Pored sažimanja ivice, moguće je izvršiti i *stapanje ivica* koje predstavlja operaciju inveznu operaciji podele ivice i *brisanje temena* koje predstavlja operaciju inverznu podeli trougla (slika 12.25).

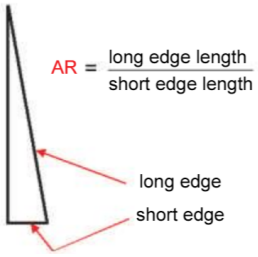


Slika 12.25: Operacija brisanja temena.

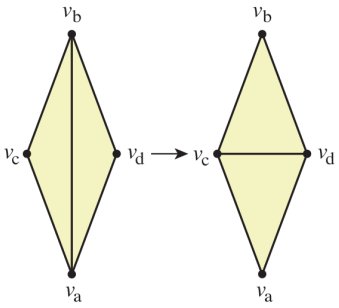
Prilikom različitih operacija nad mrežom trouglova, mreža se može deformisati, a pojedinačni trouglovi postati jako dugi i uski. Drugim rečima, odnos dužine najduže i najkraće stranice trougla (eng. aspect ratio[1]) može postati veliki (slika 12.26). Idealna situacija je kada je ta vrednost jednaka 1, što je slučaj kod jednakostraničnog trougla.

Ulepšavanje mreže (eng. mesh beautification) obuhvata operacije nad mrežom kojima se nastoji da trouglovi mreže postanu što bliskiji jednakostraničnim trouglovima i da se postignu još neka poželjna svojstva mreže (na primer, da broj suseda svakog temena bude blizak broju 6). Operacija *zamene ivice* (eng. edge swap) pomaže da se dva dugačka i uska trougla transformišu u dva skoro jednakostranična trougla (slika 12.27).

[1]Postoji više različitih definicija pojma aspect ratio trougla.



Slika 12.26: Računanje količnika dužine najduže i najkraće stranice trougla.



Slika 12.27: Operacija zamene ivice.

12.4 Pitanja

12.1 Zašto se prilikom modelovanja figura mrežama poligona, za primitive najčešće biraju trouglovi, a ne poligoni sa većim brojem stranica?

12.2 Kakve objekte je moguće precizno opisati mrežama poligona, a kakve nije?

12.3 Koje su osobine idealne mreže trouglova?

12.4 Šta se postiže operacijama kojima se vrši unapređivanje mreže?

12.5 Opisati operaciju podele trougla, podele ivice i podele temena.

12.6 Opisati operaciju sažimanja ivice. Kojoj operaciji je ova operacija inverzna?

12.7 Iz kog razloga se vrši operacija zamene ivice? Kakva mreža se njome dobija?

12.8 Na koji način su opisani poligoni u eksplicitnoj reprezentaciji mreže?

12.9 Koja je razlika između reprezentacije sa pokazivačima na liste indeksa temena i reprezentacije sa pokazivačima na liste indeksa ivica?

12.10 U kojim reprezentacijama se svaka ivica mreže iscrtava dva puta? (a) u eksplicitnoj reprezentaciji (b) u reprezentaciji sa pokazivačima na liste indeksa temena (d) u reprezentaciji sa pokazivačima na liste indeksa ivica

12.11 Opisati winged-edge strukturu podataka.

12.12 Šta se zadaje geometrijom mreže, a šta njenom topologijom?

12.13 Koja se reprezentacija koristi za predstavljanje mreže u trodimenzionom prostoru?

12.14 Kada za dvodimenzionu mrežu kažemo da je mreža mnogostrukosti? Da li je kod njih dozvoljeno dodati novi trougao u mrežu? Da li je dozvoljeno obrisati neki trougao iz mreže?

12.15 Za koju mrežu u trodimenzionom prostoru kažemo da je mreža mnogostrukosti sa granicom?

12.16 Čemu je jednaka vrednost granice kod (a) orijentisanih mreža mnogostrukosti bez granice (b) orijentisanih mreža mnogostrukosti sa granicom?