# Računarska grafika

*Beleške za predavanja*

Predrag Janičić
Matematički fakultet, Beograd
email: janicic@matf.bg.ac.rs
URL: www.matf.bg.ac.rs/~janicic

Autor:
dr Predrag Janičić, vanredni profesor Matematičkog fakulteta u Beogradu

RAČUNARSKA GRAFIKA

# Predgovor

Ovo je materijal koji prati predavanja iz predmeta *Računarska grafika* koji sam koristio akademskih godina od 2002/03 do 2004/05 i od 2007/08 do 2013/14. Funkcija materijala je da olakša praćenje predavanja i da služi kao podsetnik tokom pripremanja ispita. On ne mož da zameni pohađanje nastave i korišćenje druge literature.

Ovaj materijal zasnovan je prvenstveno na knjizi: Foley, van Dam, Feiner, Hughes: *Computer Graphics: principles and practice* i njenim novijim izdanjima, ali i mnogim drugim izvorima.

Zahvaljujem mnogobrojnim studentima koji su svojim pitanjima i sugestijama na predavanjima uticali na ovaj materijal. Dragocenu pomoć pružili su mi studenti koji su mi ukazali na greške načinjene u prethodnim verzijama materijala, a posebno Ivan Petrović.

Predrag Janičić

email: janicic@matf.bg.ac.rs
url: http://www.matf.bg.ac.rs/~janicic

Beograd, avgust 2014.

# Sadržaj

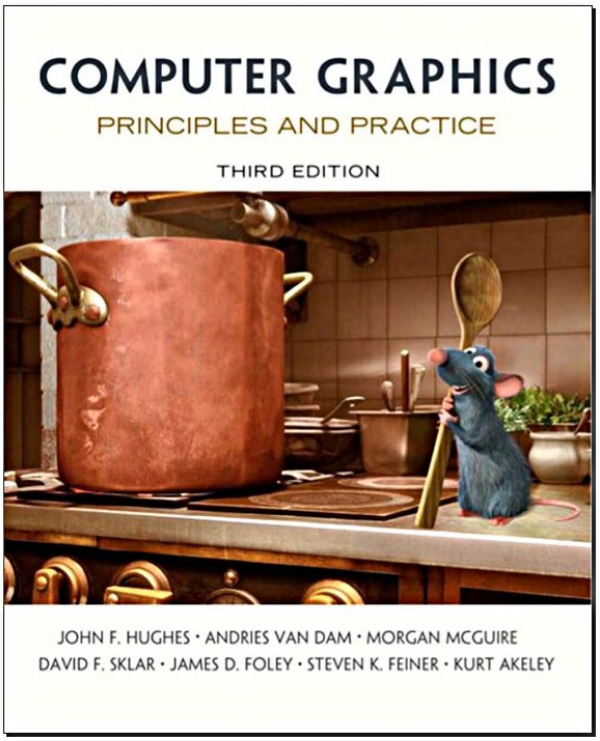
*Glava* 1

# Preporučena dodatna literatura

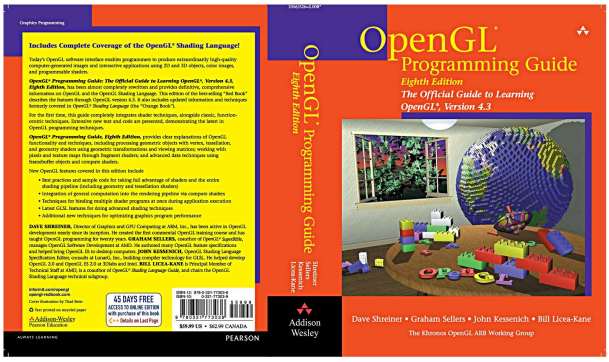
- Foley, van Dam, Feiner, Hughes: *Computer Graphics: Principles and Practice*; 2nd edition, Addison-Wesley, 1995.

  Najnovije izdanje:

  Hughes, van Dam, McGuire, Sklar, Foley, Feiner, Akeley: *Computer Graphics: Principles and Practice*; 3rd edition, Addison-Wesley, 2013.




- OpenGL Programming Guide ("Red book")
  http://www.opengl-redbook.com




- Rogers: *Procedural Elements for Computers Graphics*, McGraw-Hill, 1997.

Glava 2

# Uvod

*Računarska grafika* je oblast računarstva koja se bavi kreiranjem i obradom digitalnih vizualnih sadržaja. Ti vizualni sadržaji (na primer, slike, filmovi, simulacije) mogu da prikazuju stvarne objekte (uključujući stvarne objekte koji se ne mogu videti u stvarnom svetu - na primer, atome) ili zamišljene objekte. Računarska grafika bavi se prikazima i na osnovu dvodimenzionalnih i na osnovu trodimenzionalnih modela. Osnovne poddiscipline računarske grafike su:

**Modelovanje** se bavi opisivanjem ravanskih figura i tela (na primer, mrežnim modelima);

**Renderovanje** je proces kreiranja digitalne slike na osnovu (dvodimenzionalnog ili trodimenzionalnog) modela i (realističnog ili nerealističnog) modela ponašanje svetlosti;

**Animacije** su nizovi kreiranih slika koje, kada su prikazane brzo jedna za drugom, daju utisak glatkog kretanja.

**Obrada slika** (image processing/picture analysis) bavi se zapisivanjem i obradom slika, rekonstrukcijom dvodimenzionalnih ili trodimenzionalnih objekata na osnovu njihovih slika. Obrada slika uključuje primene u satelitskim snimanjima, medicini, kosmičkim istraživanjima, robotici, prepoznavanju slova (OCR) itd.

Kreiranje i analiziranje slika slika imaju sve više zajedničkih tačaka i često je teško povući jasnu granicu između njih (na primer, programi kao PhotoShop omogućavaju selekciju dela slike po nekom kriterijumu, a zatim obradu tog dela na neki zadati način).

*Računarska geometrija* je deo matematike koji se bavi algoritamskim rešavanjem nekih geometrijskih problema (npr. određivanje konveksnog omotača). Algoritmi računarske grafike sintetizuju slike, dok algoritmi računarske geometrije konstruišu i analiziraju geometrijske objekte.

Računarska grafika blisko je povezana i sa oblašću *vizualizacije*, ali to su ipak dve zasebne oblasti.

## 2.1 Kratka istorija računarske grafike

- Do 1950: izlaz na linijske štampače
- MIT, 1950: CRT (cathode ray tube) izlaz;
- '50-ih: ubrzo nakon pojave računara – štampanje na ploterima i prikaz na katodnim cevima (vektorska grafika);
- SAGE air-defence system, sredina 50-ih: komandni i kontrolni CRT sa korišćenjem light-pen-a;
- Ivan Sutherland, 1963: osnove interaktivnog grafičkkog interfejsa
- 1976: jedna od prvih realističnih animacija bila je u filmu *Futureworld*, i ona je uključivala animaciju ljudskog lica i ruke (Ed Catmull i Fred Parke, University of Utah);
- do '80-ih: mala, nerazvijena oblast (skup hardver, veoma skupi ili veoma komplikovani alati za grafiku);
- nakon '80-ih: računarskoj grafici naglo rastu dostupnost, značaj i popularnost zahvaljujući mikroračunarima i PC računarima; koncept *raster grafike* je omogućio novi razvoj računarske grafike.

## 2.2 Primene računarske grafike

**Podele primena računarske grafike:**

- prema tipu (dimenzionalnosti) i vrsti slike: 2D crtež, 2D slika u nijansama sive, 2D kolor slika, 3D mrežni model, 3D kolor slika sa senkama i drugim efektima
- prema nivou interakcije korisnika (npr. vektorizovan oblik ili ne)
- prema ulozi slike: da li ona faza ka završnoj slici (npr. u CAD/CAM sistemima) ili je ona završna slika
- prema odnosu između više objekata i slika

**Primeri korišćenja računarske grafike:**

- grafički korisnički interfejsi (prirodnija i lakša komunikacija sa računarom paradigme *point and click* itd; grafička manipulacija sve više zamenjuje kucanje komandi (u svim tipovima aplikacija)
- interaktivna izrada crteža u biznisu, nauci i tehnologiji (vizuelizacija je neophodna za razumevanje i interpretiranje podataka velikog obima i kompleksnih podataka).
- elektronsko izdavaštvo
- „Computer Aided Design“ - CAD (i CAM) sistemi
- Naučne simulacije i vizualizacije, kartografija i sl.
- Simulacije i animacije u zabavne svrhe
- Industrijski dizajn, umetnost i sl.

*Glava 3*

# Hardver i softver za računarsku grafiku

## 3.1 Izlazne tehnologije (tehnologija za prikaz)

Ekrani, štampači, ploteri, itd.

### 3.1.1 Adresivost i rezolucija

**Adresivost** (*addressability*) je broj pojedinačnih (ne nužno razlučivih) tačaka po inču koje mogu biti kreirane. Može da se razlikuje horizontalna adresivost i vertikalna adresivost.

**Rezolucija** Rezolucija je broj razlučivih od strane posmatrača ili uređaja različitih linija po inču (na primer, naizmenično crnih i belih) koje uređaj može da kreira. Rezolucija ne može biti veća od adresivosti.

**Veličina tačke** (*dot size*) je prečnik jedne tačke na izlaznom uređaju.

**Rastojanje između tačka** (*Interdot distance*) Obično je poželjno da je veličina tačke veća od rastojanja između tačka. Treba naći pravu meru zbog glatkih prelaza i detalja. Rastojanje između središta je količnik jednog inča i adresivosti.

### 3.1.2 Ekrani

**Vektorski sistemi:** Korišćeni su tokom 1950-ih do sredine 1970-ih.

**Funkcionisanje:** Linija se dobija tako što se digitalne koordinate krajnjih tačaka transformišu u analogni napon za elektronski zrak koji pada na površinu ekrana. Ova metodologija se zove i *random scan* (linija može da spaja „bilo koje dve tačke na ekranu“).

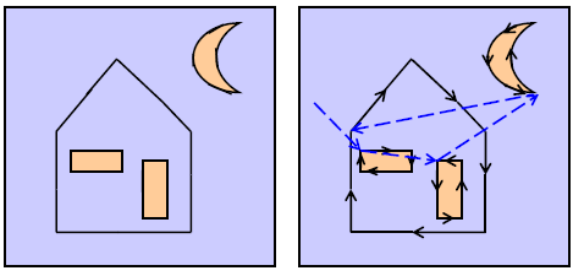
**Refresh buffer:** Bafer koji sadrži listu za prikaz.

**Refresh rate:** Slika se obnavlja obično 30 do 60 puta u sekundi (30-60 Hz).

**Kompleksnost sadržaja:** Mogu da iscrtavaju i do 100000 linija.

**Kvalitet izlaza:** Linije su glatke, ali obojene površine se teško prikazuju; pogodan samo za mrežne modele. Rezolucija je veoma velika, neki ekrani su mogli da adresiraju 4000x4000 tačaka.

**Podvrste:** *storage tube graphics display, calligraphic refresh graphics display*.



Iscrtavanje slike na vektorskom monitoru

**DVST sistemi:** 1960-e: nova generacije vektorskih sistema — *direct view storage tube* (DVST); računarska grafika postaje moguća.

**Prednosti:** Ne zahteva veliku učestalost osvežavanja (jer je slika reprezentovana raspodelom naelektrisanja na unutrašnjoj površini ekrana).

**Mane:** Prilikom izmene i najmanjeg detalja na slici, kompletna slika (raspodela naeltrisanja) mora ponovo biti kreirana.

Tektronix računar sa DVST monitorom (1970-e)

**CRT sistemi:** 1970-e: raster grafika zasnovana na televizijskoj tehnologiji i katodnim cevima (CRT, cathode ray tube). Slika je reprezentovana pikselima.[1]

**Funkcionisanje:** Elektronski top emituje zrak elektrona koji se ubrzava pozitivnim naponom ($15000V$ do $20000V$) do fosforom presvučenog

[1]Piksel, od engleskog pixel, je veštački stvorena reč. Pre njene pojave, zvanično ime jedne tačke ekrana je bilo *picture element*.

ekrana. Na putu do ekrana, mehanizmom za fokusiranje elektroni se usmeravaju u uzan zrak i usmeravaju ka konkretnoj tački na ekranu. Fosfor emituje svetlost koja opada eksponencijalno sa vremenom (što određuje karakteristika *trajnost* (persistence)), pa je potrebno sliku osvežavati (*refresh*). U sistemima sa rasterskom grafikom osvežavanje se obavlja obično bar 60 puta u sekundi (60Hz) (nezavisno od složenosti slike).

**Piksel:** nema jasno određene ivice već se za njegovu veličinu uzima prečnik oblasti gde je intenzitet emitovanja veći od 50% intenziteta u središtu oblasti. Obično je veličina tačke za monohromatske CRT visoke rezolucije oko 0.005 inča.

**Shadow mask:** kolor CRT imaju tzv. *shadow mask*. Shadow mask je tanka metalna ploča sa mnoštvom malih otvora takvih da svaki od elektronskih zrakova može da pogodi samo jednu od tri tipa otvora (za crvenu, zelenu i plavu boju). Shadow mask uzrokuje ograničenja u rezoluciji koja ne postoje kod monohromatskih CRT. Sistem u boji ima tri zraka za svaki piksel — po jedan za crvenu, zelenu i plavu boju. *Triad* je grupa tri fosforne oblasti (tačke) za crvenu, zelenu i plavu na fosfornoj površini.

**Osvežavanje:** Na ranim sistemima, osvežavanje rasterske slike (refresh rate) vršilo se po trideset puta u sekundi, kasnije obično 60 puta u sekundi (60Hz).

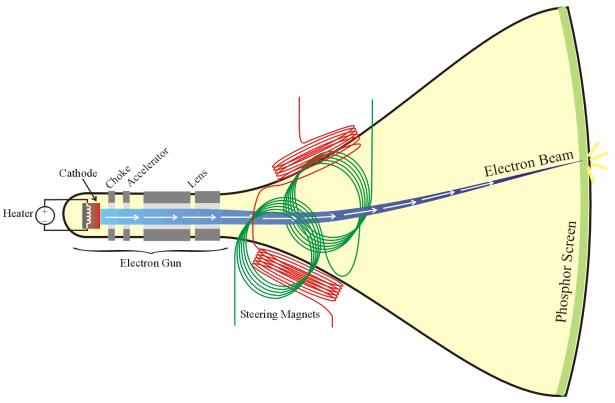
**Critical fusion frequence:** (CFF) najmanja učestalost osvežavanja slike za koju se čini da slika više ne treperi već je stabilna.

**Aspect ratio:** je fizički odnos širine i visine slike. Ovaj odnos ne mora biti isti za čitav ekran i za pojedinačni piksel. Aspect ratio 4:3 za piksel govori da je za prikaz horizontalne i vertikalne linije iste dužine potrebno 4 odnosno 3 piksela.

**Kompleksnost sadržaja:** Sadržaj slike ne utiče na brzinu prikazivanja. Kreiranje može da bude zahtevnije nego na vektorskim sistemima: na primer, za telo koje rotira u vektorskom sistemu dovoljno je preračunavati koordinate kontrolnih tačaka.

**Kvalitet izlaza:** Kose linije su „stepenaste“, ali obojene površine (popunjavanje oblasti) mogu lako da se prikazuju.

**Frame buffer:** memorijski niz u kojem je svakom pikselu pridružen prostor u kojem se čuva njegova boja. Pojava jeftinih RAM memorijskih čipova za *frame buffer* dalje je unapredila standarde u grafici.




Ilustracija rada katodne cevi

**LCD sistemi:** liquid-crystal display (LCD). Tanki, ravni displej (prikazni) uređaji. Široko rasprostranjeni od sredine 1990-ih. Kupovina LCD televizora nadmašili je kupovinu CRT televizora 2007. godine.

**Funkcionisanje:** LCD monitori 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 („pozadinsko svetlo“, backlight).

**Prednosti:** Troše veoma malo električne energije. Nema treperenja slike.

**Refresh rate:** i preko 200Hz.

**Rezolucija** LCD ekrani imaju nativnu fiksnu, rezoluciju, sa fiksnim rasterom.

***Active matrix panels*:** to su LCD ekrani kod kojih je u svaki piksel ugrađen po jedan tranzistor (koji opisuje taj piksel), te kod njih nema treperenja. Jedan od podtipova ovog tipa ekrana su TFT ekrani (*Thin Film Transistor*).

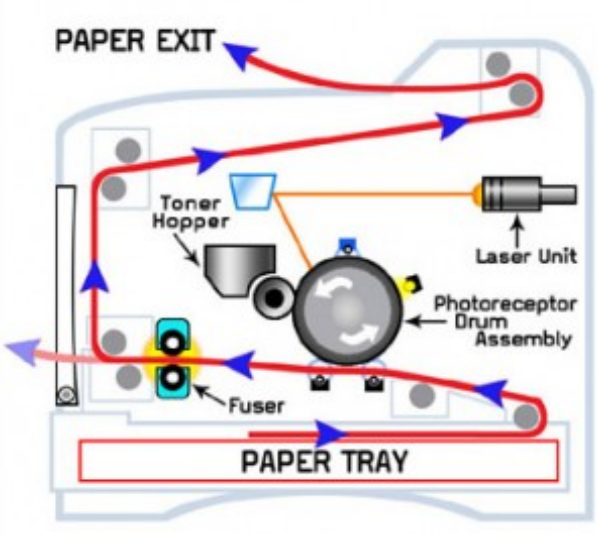
itd.

### 3.1.3 Štampači (2D)

**Matrični štampači** (*dot-matrix printers*); od 7 do 24 iglice (ili pina); one udaraju u ribon (traku) i ostavljaju trag na papiru. Adresivost može da bude i manja nego rastojanje između iglica (jer može da se štampa u dva prolaza, a pri tome je moguće pomeranje papira za dužinu manju od rastojanja između dve susedne iglice ili su iglice raspoređene pogodno u dve kolone). Postoji i kolor varijanta (sa četiri ribona).

Matrični štampač Epson LQ-570

**Laserski:** laserski zrak prelazi preko pozitivno naelektrisanog rotirajućeg doboša presvučenog selenom; oblast pogođena zrakom gubi naelektrisanje i pozitivno naelektrisanje ostaje samo tamo gde kopija treba da bude crna. Negativno naelektrisani prah tonera privlači se od strane (preostalih) pozitivno naelektrisanih oblasti. U kolor verziji ovaj postupak se ponavlja tri puta. Laserski štampači obično imaju mikroprocesore koji neposredno prihvataju i datoteke u različitim specifičnim grafičkim (kao što je PostScript – proceduralni opis dokumenta koji treba odštampati).




Ilustracija rada laserskog štampača

**Ink-jet štampači** rade na principu usmeravanja tankih mlazeva tečnog mastila na papir. Oni su najčešći korišćeni printeri u širokoj upotrebi zbog svoje niske cene, visokog (fotorealsitičnog) kvaliteta otiska, mogućnosti

za štampanje u živim bojama i jednostavne upotrebe. Najrasprostanjeniji, termalni ink-jet štampači koriste kertridže sa malim elektični zagrejanim komorama. Pri štampanju, printer pokreće struju kroz zagrejane elemente prouzrokujući u komori mini eksploziju koja stvara mehurić (bubble, otuda naziv Bubblejet za Canon-ove printere) koji dalje baca mlaz mastila na papir. Inkjet štampači su najčešće u boji i imaju zasebne komore za tri boje i crnu. Inkjet štampači imaju brojne prednosti u odnosu na druge klase: oni su tiši i precizniji nego matrični, U odnosu na laserske štampače, inkjet štampači se manje zagrevaju i jeftiniji su. Nedostaci ink-jet štampača su cena štampanja po strani (mastilo je prilično skupo) i osetljivost otiska (mastilo je najčšće rastvorivo u vodi).

**Termalni štampači** (*thermal-transfer printers*) zasnovani na korišćenju zagrevanja posebnih vrsta papira, slično kao kod faks mašina.

### 3.1.4 Ploteri

Kao što matrični štampači odgovaraju raster grafici, tako ploteri odgovaraju vektorskoj grafici. *flatbed* i *drum*, *desk-top* ploteri (u prvim se pomera samo glava plotera, a u drugim i glava plotera (preciznije, nosač glave po dužini papira, a glava po nosaču, tj. po širini papira) i ,,doboš“). Postoje i elektrostatički ploteri (bez ,,olovke“, *pen*-a — papir se na potrebnim mestima negativno naeliktriše, a onda pospe pozitivno naelektrisanim crnim tonerom).

Poslednjih godina u velikoj meri su potisnuti od strane štampača velikih formata.

### 3.1.5 Ostali hardcopy uređaji

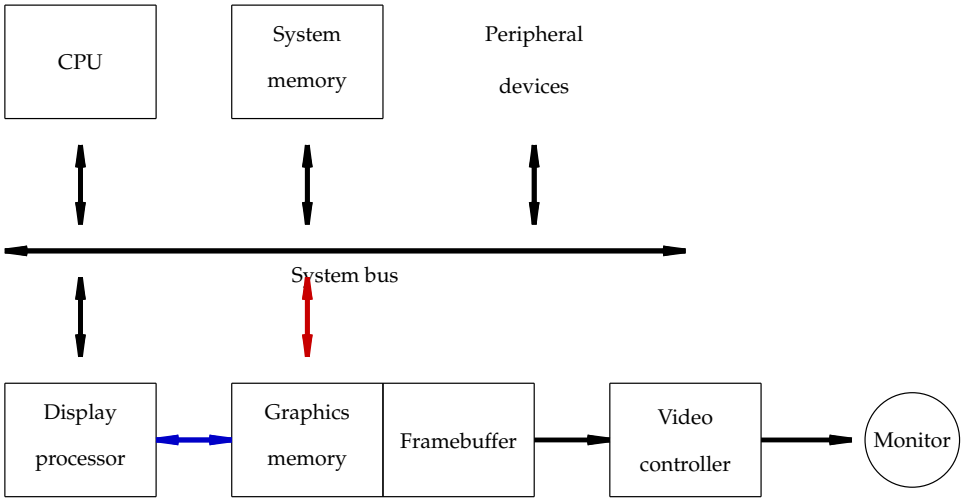
- Projektori
- 3D filmovi
- 3D projektori
- 3D štampači
- itd.

## 3.2 Ulazna tehonologija

- light pen (posebno za vektorske sisteme)
- miš (Doug Engelbart, 1968)
- paneli osetljivi na dodir - grafički tableti (npr. tablet PC)
- digitalni foto-aparati, digitalne kamere itd.
- džojstik
- trackball
- skeneri
- itd.

## 3.3 Arhitektura raster sistema za prikaz sa procesorom za prikaz

- CPU
- sistemska memorija
- procesor za prikaz ili grafički procesor (display processor ili graphics processing unit, GPU) koji kreira slike (u frejm baferu) koje će biti prikazane i komunicira sa:
  - memorijom procesora za prikaz (u kojoj su smešteni programi koje procesor izvršava prilikom iscrtavanja po frejm baferu i razni njima pridruženi podaci).
  - frejm baferom (čuva sadržaj koji se prikazuje na ekranu)
- video kontroler prolazi kroz frejm bafer i prikazuje liniju po liniju na ekranu




Slika 3.1: Jedna arhitektura raster sistema za prikaz (strelice crvene i plave boje ilustruju dve varijante)

Moderni grafički procesori, zahvaljujući svojoj paralelnoj strukturi, mogu biti efikasniji od opštenamenskih centralnih procesora i često se koriste kada je potrebno obraditi velike količine podataka paralelno (popularni jezici/sistemi: OpenCL, Nvidia CUDA).

Funkcija video kontrolera je da stalno osvežava sadržaj ekrana Postoje dva tipa:

- interlaced (osvežavaju se parne, a zatim neparne linije na npr. 30Hz)

- noninterlaced (koristi se ako cela slika može da se osvežava na više od 60Hz)

## 3.4 Softver za računarsku grafiku

Biblioteke za računarsku grafiku:

- Biblioteke zavisne od platforme
- Biblioteke niskog nivoa
- Biblioteke visokog nivoa

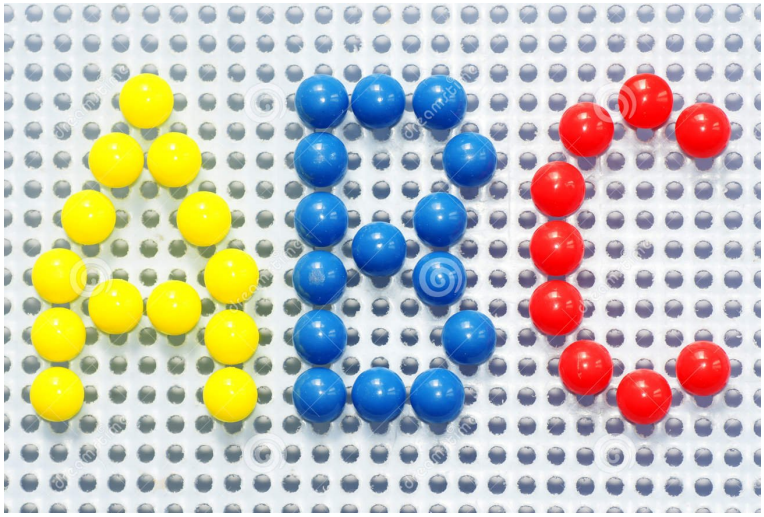
OpenGL (Open Graphics Library) je aplikacijski programski interfejs za kreiranje 2D i 3D digitalnih slika za različite programske jezike i različite platforme.

Glava 4

# Algoritmi za crtanje 2D primitiva

Opšti zadatak: crtanje na rasterskim sistemima (tj. *rasterizacija* osnovnih geometrijskih figura (duž, krug, elipsa). (Na vektorskim sistemima i u vektorskim formatima – ovaj problem ne postoji.)

Pikseli su reprezentovani ili kao krugovi sa središtima koja su čvorovi celobrojne mreže ili kao kvadrati određeni celobrojnom mrežom. U narednim algoritmima, smatraće se da su pikseli reprezentovani kao krugovi sa središtima koja su čvorovi celobrojne mreže.



## 4.1 Crtanje duži (scan converting line)

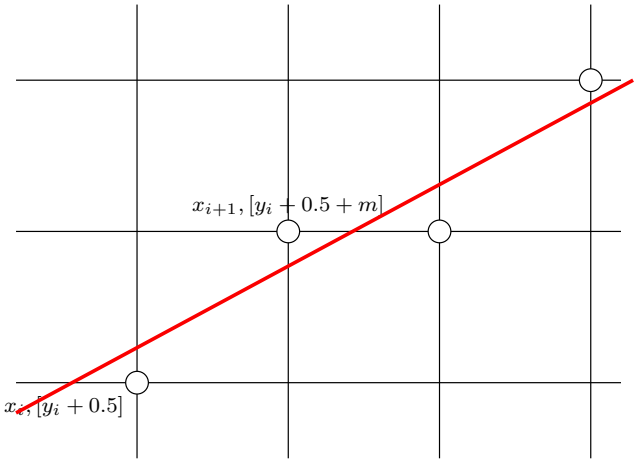
Zahtevi:

- Niz piksela treba da bude što bliže idealnoj liniji.
- Crtanje treba da bude što je moguće brže.
- Dodatni kriterijumi:

- Deblje linije i linije nacrtane određenim stilom.
- Sve linije treba da budu iste osvetljenosti i sve tačke svake linije treba da budu iste osvetljenosti (ili da tako izgledaju) bez obzira na njihov nagib i dužinu.
- Varijante za crtanje linije u tehnici antialiasing.

### 4.1.1 Osnovni zadatak

- Debljina duži 1.
- Jednobitna slika (svaki piksel može biti samo uključen ili isključen);
- Crtanje tačke je primitiva.
- Horizontalne, vertikalne i duži koeficijenta 1 ili -1 crtaju se trivijalno; problem je nacrtati ostale duži.
- Razmatrati samo duži koeficijenta $m$ takvih da je $|m| < 1$; ostali slučajevi rešavaju se simetrično.
- Za duži koeficijenta između -1 i 1, u svakoj koloni treba da bude označen tačno jedan piksel; za duži van tog opsega, u svakoj vrsti treba da bude označen bar jedan piksel.
- Različiti kriterijumi ocene kvaliteta (rastojanja od idealne duži).




### 4.1.2 Osnovni inkrementalni algoritam

- Koeficijent $m$:
$$m = \Delta y / \Delta x$$
- Gruba sila:
$$y_i = mx_i + B,$$
za $i = 1, 2, 3, \ldots$ i pri čemu se $x_i$ inkrementira za po 1.

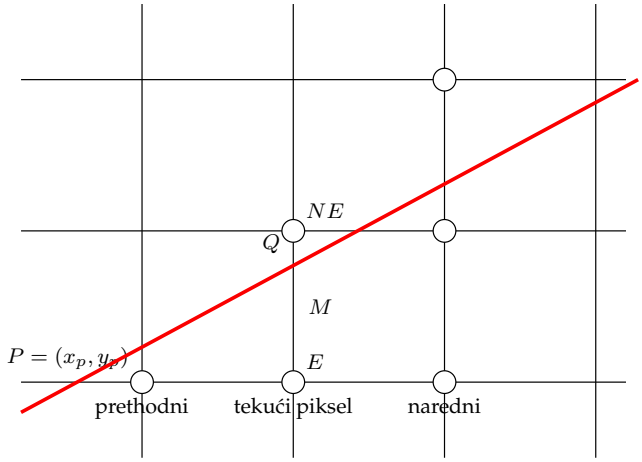
```
procedure Line (x0, y0, x1, y1 : integer);
var
   x : integer;
   dx, dy, y, m : real;

begin
   dx := x1 - x0;
   dy := y1 - y0;
   m := dy/dx;
   y := y0;
   for x := x0 to x1 do
     begin
       SetPixel(x, Round(y));
       y := y+m
     end
end.
```

Slika 4.1: Algoritam za crtanje duži

- Treba osvetliti tačku
$$(x_i, [y_i + 0.5])$$
Tačno (koliko je to moguće), ali neefikasno zbog toga što svaki korak zahteva množenje, sabiranje i zaokruživanje ($y$ i $x$ su promenljive realnog tipa)
- Gornja veza se može izraziti i na sledeći način:
$$y_{i+1} = mx_{i+1} + B = m(x_i + \delta x) + B = mx_i + m\delta x + B = y_i + m\delta x$$
- Ako je $\delta x = 1$, onda je
$$y_{i+1} = y_i + m$$
čime je eliminisano množenje.
- Ovo ilustruje prirodu inkrementalnih algoritama — u svakom koraku izračunavanja pravimo na osnovu prethodnog koraka.
- Primetimo da se u ovom pristupu ne treba eksplicitno starati o slobodnom članu $B$; za početnu tačku uzima se tačka sa najmanjom $x$ koordinatom.
- Ovaj algoritam se često zove *DDA* (digital differential algorithm); DDA je mehanički uređaj koji rešava diferencijalne jednačine numeričkim metodama.
- Vrednost $m$ je realna vrednost i, kao takva, ona može biti izračunata neprecizno; u uobičajenim okolnostima (relativno kratke duži) ta neprecizност nije bitna.

### 4.1.3 Midpoint algoritam za crtanje duži (varijanta Bresenhamovog algoritma)

- Ključni nedostaci osnovnog inkrementalnog algoritma su potreba zaokruživanja i to što promenljive moraju da budu realnog tipa.
- Bresenham (1965): originalno razvijen za digitalni ploter; celobrojna aritmetika, tehnika može da se koristi i za krug; može da se napravi i realna varijanta; dokazano da minimizuje rastojanja od idealne duži;
- Pitteway (1967) i Van Aken (1984) — midpoint algoritam: za duž se ponaša isto kao Bresenhamov algoritam, ali može da se uopšti na proizvoljne krive drugog reda.




- Neka je nagib između 0 i 1;
- Neka je potrebno spojiti tačke $(x_0, y_0)$ i $(x_1, y_1)$;
- $NE$=northeast; $E$=east; $M$=midpoint
- Nakon tačke $P$ treba odabrati $E$ ili $NE$ (jer je nagib između 0 i 1)
- Ako je $M$ „ispod“ prave koja sadrži duž, onda je pravoj bliža $NE$, a inače $E$.
- Neka je prava
$$F(x, y) = ax + by + c$$
- Ako je $dx = x_1 - x_0$ i $dy = y_1 - y_0$:
$$y = \frac{dy}{dx}x + B$$
i
$$F(x, y) = dy \cdot x - dx \cdot y + B \cdot dx = 0$$

- Vrednost $F(x, y)$ je jednaka $0$ za tačke na pravoj, pozitivna za tačke „ispod“ prave i negativna za tačke „iznad“ prave.
- Treba odrediti znak $F(M) = F(x_p + 1, y_p + \frac{1}{2})$
- Vrednost $d = F(x_p + 1, y_p + \frac{1}{2})$ zovemo *promenljiva odlučivanja*.
- Ako je $d < 0$, onda biramo $E$, ako je $d > 0$, onda biramo $NE$, Ako je $d = 0$, onda je svejedno i, po dogovoru, biramo $E$.
- Ako smo odabrali $E$ ili $NE$, kako odabrati sledeći piksel? U odlučivanju se koristi prethodna vrednost promenljive odlučivanja:
  - Ako smo odabrali $E$:

$$d_{new} = F(x_p + 2, y_p + \frac{1}{2}) = a(x_p + 2) + b(y_p + \frac{1}{2}) + c =$$

$$a + a(x_p + 1) + b(y_p + \frac{1}{2}) + c = a + d_{old}$$

    Tj.:

$$d_{new} = d_{old} + a = d_{old} + dy$$

  - Ako smo odabrali $NE$:

$$d_{new} = F(x_p + 2, y_p + \frac{3}{2}) = a(x_p + 2) + b(y_p + \frac{3}{2}) + c =$$

$$a + b + a(x_p + 1) + b(y_p + \frac{1}{2}) + c = a + b + d_{old}$$

    Tj.:

$$d_{new} = d_{old} + a + b = d_{old} + dy - dx$$

- Za prvu tačku $(x_0, y_0)$ važi $F(x_0, y_0) = 0$ (jer je na pravoj).
- Za prvo središte važi:

$$d_{start} = F(x_0 + 1, y_0 + \frac{1}{2}) = a(x_0 + 1) + b(y_0 + \frac{1}{2}) + c =$$

$$ax_0 + by_0 + c + a + b/2 = F(x_0, y_0) + a + b/2 = a + b/2$$

- Da bi se izbeglo deljenje sa $2$ u $d_{start}$ sve vrednosti se množe sa dva (a svi relevantni znakovi ostaju isti). Dakle:
  - $d_{start} = 2a + b = 2dy - dx$
  - ako je $d_{old} \leq 0$, onda $d_{new} = d_{old} + 2dy$
  - ako je $d_{old} > 0$, onda je $d_{new} = d_{old} + 2dy - 2dx$

### 4.1.4 Crtanje duži – dodatna pitanja

- Poredak tačaka: duži $P_0P_1$ i $P_1P_0$ mora da se crtaju isto (tj. da se sastoje od istih piksela) (kada se ide „sleva nadesno“ bira se $E$ za $d = 0$, a kada se ide „zdesna nalevo“ bira se $SW$ (south-west)). Može i jednostavno da se poredak tačaka svede na poredak „sleva nadesno“ ali to ne daje ispravan rezultat ako se koriste stilovi.
- Intenzitet boja: razmotriti duži od deset tačaka sa nagibom $0$ i nagibom $1$. Obe imaju po deset piksela, ali je ova druga duža $\sqrt{2}$ puta od prve. Toliko je i intenzitet puta veći za prvu nego za drugu. Ako su na uređaju raspoložive samo dve boje, to ne može da se popravi. Ako su na raspolaganju nijanse boje, onda one mogu da se upotrebe i biraju u zavisnosti od nagiba duži. U te svrhe koristi se i tehnika antialiasing.

## 4.2 Crtanje kruga

Zahtevi:

- Slično kao za duži: da niz piksela bude što bliže idealnom krugu, da crtanje bude što je moguće brže, da nacrtani krug bude „neprekidan“ itd.
- Posebno se razmatra varijanta za crtanje duži u tehnici antialiasing.

### 4.2.1 Osnovni zadatak

- Debljina kruga 1
- Jednobitna slika
- Crtanje tačke je primitiva
- Pretpostavlja se da je središte kruga tačka (tj. piksel) $(0, 0)$; algoritam se može trivijalno uopštiti tako da radi i za druge slučajeve
- Dovoljno je odrediti sve tačke kruga u jednom kvadrantu; na osnovu tih tačaka se određuju i ostale tačke (na osnovu simetrije), mada u realnim implementacijama može da se, zbog efikanosti, sličan kod navede više puta.

### 4.2.2 Naivni algoritam

- Iz jednačine kruga:
$$x^2 + y^2 = R^2$$
sledi:
$$y = \pm\sqrt{R^2 - x^2}$$
- za $i = 1, 2, 3, \ldots$, pri čemu se $x_i$ inkrementira za po 1, treba osvetliti tačku
$$(x_i, [\sqrt{R^2 - x_i^2} + 0.5])$$
Tačno (koliko je to moguće), ali neefikasno.

- sličan neefikasan metod: crtati tačke

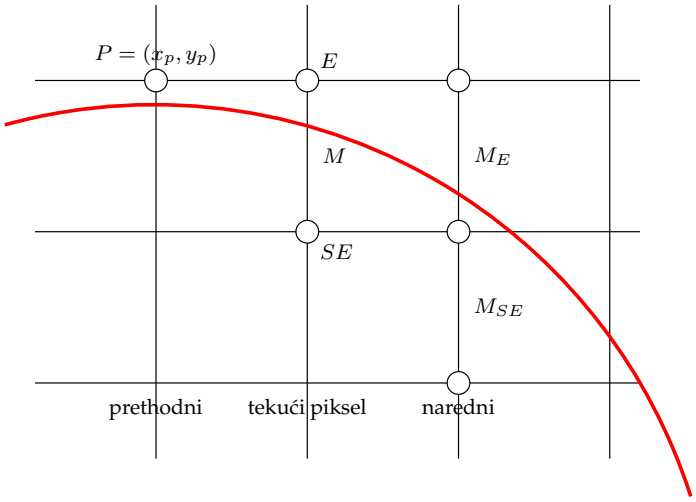
$$([R\cos\varphi_i + 0.5], [R\sin\varphi_i + 0.5])$$

za $\varphi_i = 1°, 2°, 3°, \ldots, 90°$.

- Kako vrednost $x$ raste, povećavaju se praznine između tačaka; to je lako rešiti korišćenjem i simetrije kruga po još jednoj osi (pravoj $y = x$).

### 4.2.3 Bresenham (midpoint) inkrementalni algoritam

- Bresenham (1977): originalno razvijen za digitalni ploter;
- Midpoint algoritam: varijanta Bresenhamovog algoritma.
- Razmatra se samo osmina kruga
- Vrednost $x$ ide od $0$ do $R/\sqrt{2}$
- Osnovna ideja slična je ideji za crtanje duži: u svakom koraku treba odabrati jednu od dve moguće tačke




- Vrednost $F(x, y) = x^2 + y^2 - R^2$ je jednaka $0$ u tačkama koje pripadaju krugu, pozitivna u spoljašnjosti kruga i negativna u unutrašnjosti kruga.
- Ako je tačka $M$ u unutrašnjosti kruga, onda je tačka $E$ bliža krugu nego tačka $SE$. Ako je tačka $M$ u spoljašnjosti kruga, onda je tačka $SE$ bliža krugu.
- Promenljiva odlučivanja jednaka je:

$$d_{old} = F(x_p + 1, y_p - \frac{1}{2}) = (x_p + 1)^2 + (y_p - \frac{1}{2})^2 - R^2$$

- Ako je $d_{old} < 0$, tačka $E$ je izabrana, ako je $d_{old} > 0$, tačka $SE$ je izabrana. Ako je $d_{old} = 0$, onda je izbor tačke stvar dogovora i bira se tačka $SE$.

- Ako je $d_{old} < 0$, tačka $E$ je izabrana i važi

$$d_{new} = F(x_p + 2, y_p - \frac{1}{2}) = (x_p + 2)^2 + (y_p - \frac{1}{2})^2 - R^2 =$$
$$= d_{old} + (2x_p + 3)$$

- Ako je $d_{old} \geq 0$, tačka $SE$ je izabrana i važi

$$d_{new} = F(x_p + 2, y_p - \frac{3}{2}) = (x_p + 2)^2 + (y_p - \frac{3}{2})^2 - R^2 =$$
$$= d_{old} + (2x_p - 2y_p + 5)$$

- U slučaju $d_{old} < 0$ koristi se $\Delta_E = 2x_p + 3$, a u slučaju $d_{old} \geq 0$ koristi se $\Delta_{SE} = 2x_p - 2y_p + 5$. Primetimo da obe ove razlike zavise od koordinata tačke $P$.

```
procedure MidpointCircle (radius : integer);
var
   x, y, d : integer;
begin
   x:=0;
   y:=radius;
   d:=1-radius;
   SetPixel(x, y);

   while (y>x) do
      begin
         if d<0 then    { select E }
            begin
               d:=d+2*x+3;
               x:=x+1;
            end
         else       { select SE }
            begin
               d:=d+2*(x-y)+5;
               x:=x+1;
               y:=y-1;
            end;
         SetPixel(x, y);
      end
end.
```

Slika 4.2: Algoritam za crtanje kruga

- Početni uslov (za celobrojni poluprečnik):

$$F(0, R) = 0$$

$$F(1, R - \frac{1}{2}) = 1 + (R^2 - R + \frac{1}{4}) - R^2 = \frac{5}{4} - R$$

- Nije dobro to što prva vrednost za $d$ nije celobrojna
- koristićemo zamenu $h = d - \frac{1}{4}$: tada je inicijalna vrednost $h = 1 - R$, a poređenje $d < 0$ postaje $h < -\frac{1}{4}$. Međutim, kako je inicijalna vrednost celobrojna i kako su vrednosti koje se dodaju ($\Delta_E$ i $\Delta_{SE}$) celobrojne, ovaj uslov može da se zameni sa $h < 0$; tako uvedenu promenljivu $h$ na kraju ipak označimo sa $d$.

### 4.2.4 Bresenham inkrementalni algoritam — unapređena verzija

- Vrednosti $\Delta_E$ i $\Delta_{SE}$ su linearne i za njihovo izračunavanje takođe možemo koristiti tehniku inkrementalnog uvećavanja. Tako će dodatno biti smanjen broj sabiranja/oduzimanja.
- Ako je u jednom koraku izabrana tačka $E$, onda se tačka izračunavanja, referentna tačka pomera iz $(x_p, y_p)$ u $(x_p + 1, y_p)$. Vrednost $\Delta_{Eold}$ u $(x_p, y_p)$ je jednaka $2x_p + 3$. Dakle, vrednost $\Delta_{Enew}$ (u tački $(x_p + 1, y_p)$) je jednaka
$$\Delta_{Enew} = 2(x_p + 1) + 3 = \Delta_{Eold} + 2$$
Dodatno, vrednost $\Delta_{SEold}$ u $(x_p, y_p)$ jednaka je $2x_p - 2y_p + 5$ a vrednost $\Delta_{SEnew}$ (u tački $(x_p + 1, y_p)$) je jednaka
$$\Delta_{SEnew} = 2(x_p + 1) - 2y_p + 5 = \Delta_{SEold} + 2$$
(ove vrednosti zovemo *razlike drugog reda*)
- Analogno, ako je izabrana tačka $SE$, onda se tačka izračunavanja, referentna tačka pomera iz $(x_p, y_p)$ u $(x_p+1, y_p-1)$. Vrednost $\Delta_{Eold}$ u $(x_p, y_p)$ je jednaka $2x_p + 3$. Dakle, vrednost $\Delta_{Enew}$ (u tački $(x_p + 1, y_p - 1)$) je jednaka
$$\Delta_{Enew} = 2(x_p + 1) + 3 = \Delta_{Eold} + 2$$
Dodatno, vrednost $\Delta_{SEold}$ u $(x_p, y_p)$ jednaka je $2x_p - 2y_p + 5$ a vrednost $\Delta_{SEnew}$ (u tački $(x_p + 1, y_p - 1)$) je jednaka
$$\Delta_{SEnew} = 2(x_p + 1) - 2(y_p - 1) + 5 = \Delta_{SEold} + 4$$

### 4.2.5 Crtanje kruga čije središte nije u koordinatnom početku

- Ako središte kruga nije tačka $(0, 0$ nego $(a, b)$, koristi se algoritam sličan navedenom.
- Nije isplativno svaku za svaki piksel pojedinačno dodavati $a$ i $b$ (u odnosu na piksele kruga sa središtem $(0, 0)$.
- Sve razlike prvog i drugog reda su iste (ako je poluprečnik isti) bez obzira na središte kruga.
- Razlikuje se samo početni piksel – umesto $(0, R)$, prva uključena piksel treba da bude $(a, R+b)$ a umesto uslova `y>x` treba korsitit uslov `y-b>x-a` ili efikasnije `y>x+(b-a)`.

```
procedure MidpointCircle2 (radius : integer);
var
   x, y, d, deltaE, deltaSE : integer;
begin
   x:=0;
   y:=radius;
   d:=1-radius;
   deltaE:=3;
   deltaSE:=-2*radius+5;
   SetPixel(x, y);

   while (y>x) do
      begin
         if d<0 then    { select E }
            begin
               d:=d+deltaE;
               deltaE:=deltaE+2;
               deltaSE:=deltaSE+2;
               x:=x+1;
            end
         else       { select SE }
            begin
               d:=d+deltaSE;
               deltaE:=deltaE+2;
               deltaSE:=deltaSE+4;
               x:=x+1;
               y:=y-1;
            end;
         SetPixel(x, y);
      end
end.
```

Slika 4.3: Algoritam za crtanje kruga — unapređena verzija

## 4.3 Crtanje elipse

- Slični zahtevi kao za krug.
- Razmatramo samo slučaj kada je središte elipse u koordinatnom početku i kadu su ose elipse jednake koordinatnim osama; predstavljeni algoritam se može prilagoditi opštem slučaju.
- Iz jednačine

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

dobija se jednačina

$$F(x, y) = b^2x^2 + a^2y^2 - a^2b^2$$

- Za tačke na elipsi važi $F(x, y) = 0$, za tačke u unutrašnjasnosti elipse važi $F(x, y) < 0$, a za tačke u spoljašnjasnosti elipse važi $F(x, y) > 0$.
- Predstavljeni algoritam je zasnovan na tehnici središta (odnosno Bresenhamovim idejama).
- Tačke elipse crtaju se simetrično u četiri kvadranta, pa se može razmatrati samo prvi kvadrant.

### 4.3.1 Osnove algoritma, simetrije i inkrementalne veze

- Ttačke elipse crtaju se simetrično u četiri kvadranta. Elipsa nije simetrična u odnosu na pravu $x = y$.
- Granica nije kao kod kruga prava $x = y$ nego prava koja seče elipsu u tački u kojoj tangenta ima koeficijent $-1$.
- Granični uslov (između dva regiona) je:

$$a^2(y_p - \frac{1}{2}) \leq b^2(x_p + 1)$$

- U prvom regionu bira se između tačaka $E$ i $SE$, a u drugom, bira se između tačaka $S$ i $SE$.
- U prvom regionu, ako je izabrana tačka $E$, onda je

$$d_{old} = F(x_p + 1, y_p - \frac{1}{2}) = b^2(x_p + 1)^2 + a^2(y_p - \frac{1}{2})^2 - a^2b^2$$

$$d_{new} = F(x_p + 2, y_p - \frac{1}{2}) = b^2(x_p + 2)^2 + a^2(y_p - \frac{1}{2})^2 - a^2b^2$$

$$d_{new} = d_{old} + b^2(2x_p + 3)$$

$$\Delta_E = b^2(2x_p + 3)$$

- U prvom regionu, ako je izabrana tačka $SE$, onda je

$$d_{new} = F(x_p + 2, y_p - \frac{3}{2}) = b^2(x_p + 2)^2 + a^2(y_p - \frac{3}{2})^2 - a^2b^2$$

$$d_{new} = d_{old} + b^2(2x_p + 3) + a^2(-2y_p + 2)$$

$$\Delta_{SE} = b^2(2x_p + 3) + a^2(-2y_p + 2)$$

- Slično za drugi region.
- Početni uslov:

$$F(1, b - \frac{1}{2}) = b^2 + a^2(b - \frac{1}{2})^2 - a^2b^2 = b^2 + a^2(-b + \frac{1}{4})$$

- Ako su vrednosti $a$ i $b$ celobrojne, onda se može napraviti celobrojna verzija algoritma
- Mogu se koristiti razlike drugog reda, slično kao kod kruga

```
procedure MidpointEllipse (a, b : integer);
var
   x, y : integer;
   d1, d2 : real;

begin
   x:=0;
   y:=b;
   d1:=b^2 -a^2 * b+ a^2/4;
   SetPixel(x, y);

   while (a^2*(y-1/2)>b^2*(x+1)) do
      begin
         if d1<0 then    { select E }
            begin
               d1:=d1+b^2*(2*x+3);
               x:=x+1;
            end
         else       { select SE }
            begin
               d1:=d1+b^2*(2*x+3)+a^2*(-2*y+2);
               x:=x+1;
               y:=y-1;
            end;
         SetPixel(x, y);
      end; { region 1 }

   d2:= b^2*(x+1/2)^2+ a^2*(y-1)^2 - a^2*b^2;
   while (y > 0) do
      begin
         if d2<0 then    { select SE }
            begin
               d2:=d2+b^2*(2*x+2)+a^2(-2*y+3);
               x:=x+1;
               y:=y-1;
            end
         else       { select S }
            begin
               d2:=d2+a^2(-2y+3);
               y:=y-1;
            end;
         SetPixel(x, y);
      end; { region 2 }
end.
```

Slika 4.4: Algoritam za crtanje elipse

Glava 5

# Osnovni 2D algoritmi

## 5.1 Popunjavanje poligona

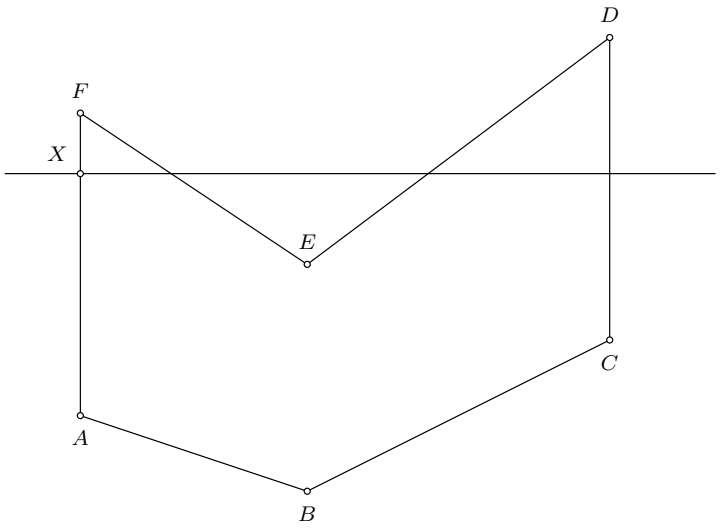
- Zadatak: obojiti sve pixele koji se nalaze u unutrašnjosti poligona istom bojom (ili istim uzorkom).
- Voditi računa o tome da susedni poligoni budu popunjeni smisleno.

### 5.1.1 Naivna rešenja

- Za svaku tačku na slici (pojedinačno) proveriti da li pripada unutrašnjosti poligona ili ne; postoje efikasni algoritmi za proveravanje da li tačka pripada unutrašnjosti poligona.
- Najpre se određuje najmanji pravougaonik (sa stranicama paralelnim koordinatnim osama) koji sadrži dati poligon, a zatim se za svaku tačku tog pravougaonika proverava da li pripada unutrašnjosti poligona ili ne;
- Naivni/opšti fill (ili *flood-fill*) algoritam (za popunjavanje proizvoljne konture – ne nužno poligona):

```
procedure Fill (x, y : integer);
var
   x, y : integer;
begin
   if BelongsToArea(x,y) { tj. ne pripada rubu oblasti }
       then SetPixel(x,y);
   if BelongsToArea(x-1,y)
        then Fill(x-1,y);
   if BelongsToArea(x+1,y)
        then Fill(x+1,y);
   if BelongsToArea(x,y-1)
        then Fill(x,y-1);
   if BelongsToArea(x,y+1)
        then Fill(x,y+1)
end.
```

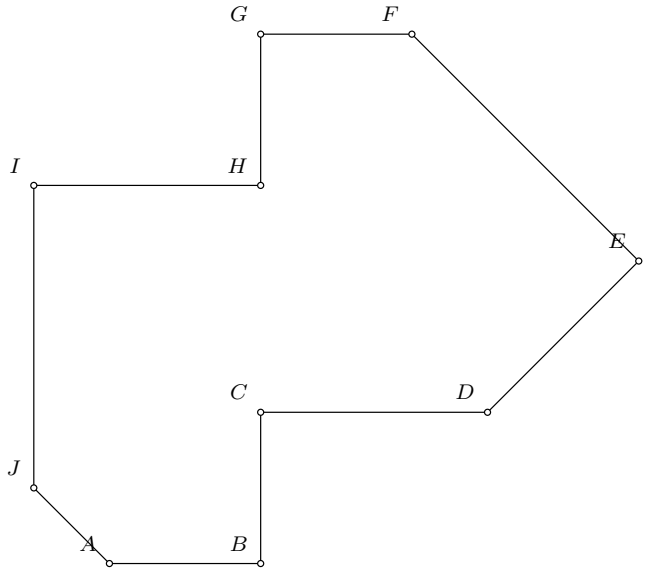
### 5.1.2 Scan popunjavanje poligona




- Na svakoj scan liniji treba odrediti tačke koje pripadaju stranicama poligona; da li je tačka na liniji određuje se na osnovu parnosti — inicijalno je brojač paran (tj. 0) i njegova parnost se menja svaki put prilikom nailaska na tačku neke stranice. Kod tačaka sa neparnim stanjem treba započeti bojenje, a kod tačaka sa parnim treba ga zaustaviti.
- U računanju parnosti, za jednu duž računa se tačka preseka sa najmanjom $y$ koordinatom ($y_{min}$), ali ne i tačka sa najvećom $y$ koordinatom ($y_{max}$). Za horizontalne stranice ne treba brojati nijednu tačku preseka.
- Ovakav algoritam ne boji „gornje“ i „desne“ „krajnje“ tačke; on ne boji neke potrebne tačke (tačke na stranicama ionako je lako obojiti), a sigurno ne boji nijednu pogrešnu tačku.
- *Sliver:* neke scan linije će imati samo jedan ili nijedan piksel; taj problem može se rešavati tehnikom antialiasing.

Određivanje preseka:

- Određivanje preseka scan linije sa svim stranicama poligona nije racionalno (najčeše samo nekoliko stranica seče scan liniju)
- Najčešće stranice koje seče $n$-ta scan linija seče i $n + 1$-a scan linija.
- Određivanje preseka scan linije sa stranicama poligona mora biti urađeno efikasno
- Određivanje preseka na svakoj stranici vrši se na bazi midpoint algoritma (odnosno njegove ideje)
- Midpoint algoritam treba modifikovati tako da se uvek biraju tačke koje pripadaju unutrašnjosti poligona.

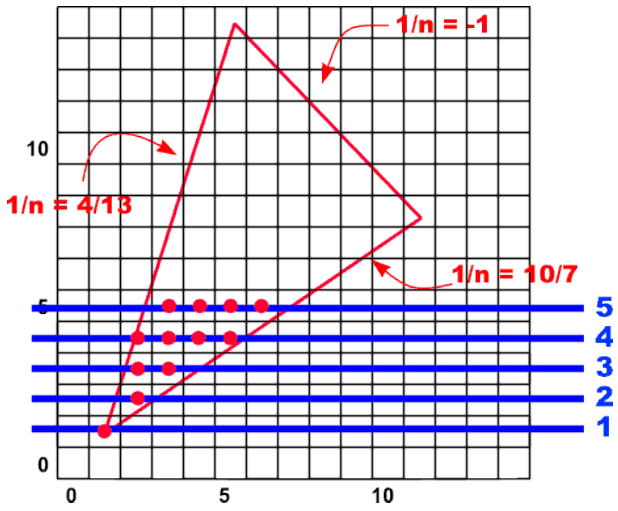



- Na osnovu ideje midpoint algoritma, može se odrediti novi presek sa stranicom na osnovu starog preseka:

$$x_{i+1} = x_i + 1/m$$

gde je $m$ koeficijent prave koja sadrži stranicu.

- Treba modifikovati navedenu vezu tako da koristi celobrojnu aritmetiku
- Potrebno je uraditi pogodno zaokruživanje kako bi bile obojene samo tačke u unutrašnjosti poligona.
- Razmotrimo „levu“ stranicu (parnost pre ovog preseka je $0$) sa koeficijentom većim od 1.
- $x_{i+1} = x_i + 1/m$
- $m = (y_{max} - y_{min})/(x_{max} - x_{min})$
- $x_{i+1} = x_i + (x_{max} - x_{min})/(y_{max} - y_{min})$
- Posebno razmatramo celi i razlomljeni deo vrednosti $(x_{max} - x_{min})/(y_{max} - y_{min})$. Kada razlomljeni deo pređe $1$, onda će celi deo da bude inkrementiran. Kada je razlomljeni deo jednak $0$, onda možemo da obojimo pixel $(x, y)$, a kada je razlomljeni deo različit od $0$, onda je potrebno izvršiti zaokruživanje takvo da $x$ pripada unutrašnjosti poligona. Kada razlomljeni deo pređe $1$, onda $x$ povećavamo za $1$ (i pomeramo se jedan piksel udesno) i razlomljeni deo smanjujemo za $1$.
- Opisani postupak može se svesti na celobrojni račun.




```
procedure LeftEdgeScan(xmin, ymin, xmax, ymax : integer};
begin
   x:=xmin;
   y:=ymin;
   brojilac:=xmax-xmin;
   imenilac:=ymax-ymin;
   increment:=imenilac;
   for y:=ymin to ymax do
      begin
         SetPixel(x,y);
         incerement := increment+brojilac;
         if increment>imenilac then
            begin  { prekoracenje, zaokruzi na sledeci }
                   { pixel i smanji inkrement }
               x:=x+1;
               increment := increment - imenilac;
            end;
      end
end.
```

Slika 5.1: Algoritam LeftEdgeScan (za koeficijent prave veći od 1)

Praktična implementacija:

- Tačke preseka se čuvaju u specijalnoj strukturi podataka koja čuva podatke o *aktivnim stranicama* – $A$. Aktivne stranice su one koje se seku sa scan linijom.

1. Polazi se od inicijalne scan prave, koja ima $y$ vrednost kao najmanja $y$ vrednost svih temena.
2. Inicijelno je skup $A$ prazan. Inicijelno skup $S$ sadrži sve stranice (sortirane po najmanjoj $y$ vrednosti).

3. Sve dok skup aktivnih stranica nije prazan ili skup stranica $S$ nije prazan, radi sledeće:

   a) Prebaci iz skupa $S$ u skup $A$ one stranice čija minimalna $y$ vrednost je jednaka $y$ vrednosti scan prave.

   b) Sortiraj stranice u skupu $A$ rastuće prema $x$ koordinati preseka sa scan linijom.

   c) Oboji sve piksele na scan liniji izmedju neparnih i parnih preseka.

   d) Izbaci iz skupa $A$ one stranice čija maksimalna $y$ vrednost je jednaka $y$ vrednosti scan prave.

   e) Uvećan $y$ vrednost scan linije za $1$.

   f) Ažuriraj tačke preseka kao u algoritmu `LeftEdgeScan`

## 5.2 Kliping/seckanje linija

Zadatak:

- Za dati pravougaonik, nacrtati samo duži i delove duži koji mu pripadaju
- Ako su koordinate donjeg levog ugla pravougaonika $(x_{min}, y_{min})$ a gornjeg desnog $(x_{max}, y_{max})$, onda treba da budu nacrtane samo tačke $(x, y)$ za koje važi

$$x_{min} \leq x \leq x_{max}, \quad y_{min} \leq y \leq y_{max}$$

- rezultat je uvek jedna duž
- kliping krugova i elipsi može da da više lukova
- kliping za krugove i elipse može da se svede na kliping (dovoljno malih) duži.

### 5.2.1 Naivno rešenje

- Za svaku duž ispitati da li njena temena pripadaju pravougaoniku, ako da, onda prihvatiti celu tu duž; ako ne, onda odrediti preseke sa svim pravama koje određuju stranice pravouganika, ispitati raspored itd.
- Parametarski oblik za duž je pogodan za određivanje preseka duži:

$$x = x_0 + t(x_1 - x_0), \quad y = y_0 + t(y_1 - y_0)$$

  gde je $t \in [0, 1]$.

- Ovaj pristup ipak zahteva mnogo izračunavanja i neefikasan je.

### 5.2.2 Cohen-Sutherland-ov algoritam

- Ako temena duži pripadaju pravougaoniku, onda prihvatiti celu tu duž.
- Ako su obe $x$ koordinate temena duži manje od $x_{min}$, onda odbaciti celu tu duž (analogno za $x_{max}$, $y_{min}$, $y_{max}$).
- Ako je jedno teme duži van pravougaonika, onda odbaciti deo duži do pravougaonika i nastaviti analogno.
- Kodovi koji se pridružuju tačkama:

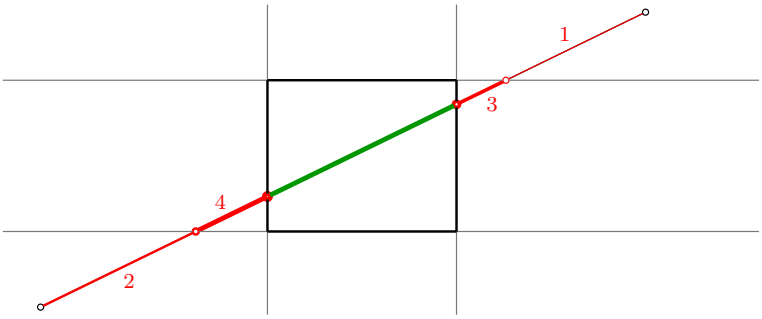
  Prvi bit (najmanje težine): „iznad gornje stranice“

  Drugi bit (najmanje težine): „ispod donje stranice“

  Treći bit (najmanje težine): „desno od desne stranice“

  Četvrti bit (najmanje težine): „levo od leve stranice“

| | | |
|---|---|---|
| 0101 | 0001 | 1001 |
| 0100 | 0000 | 1010 |
| 0110 | 0010 | 1010 |




```
#define TOP 1
#define BOTTOM 2
#define LEFT 4
#define RIGHT 8

char CompOutCode(double x, double y,
 double xmin, double ymin, double xmax, double ymax) {
    char code = 0;
    if (y>ymax)
code |= TOP;
    else if (y<ymin)
       code |= BOTTOM;
    if (x>xmax)
       code |= RIGHT;
    else if (x<xmin)
       code |= LEFT;
    return code;
}
```

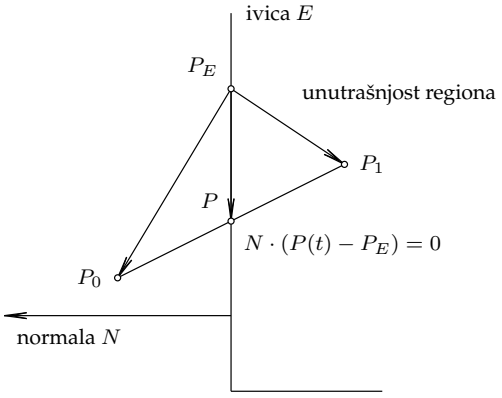
```
void CohenSutherlandLineClipAndDraw(
      double x0, double y0, double x1, double y1,
      double xmin, double ymin, double xmax, double ymax)  {
   bool accept = false, done = false;
   char pointCode0, pointCode1, pointOut;
   double x, y;

   pointCode0 = CompOutCode(x0,y0,xmin,ymin,xmax,ymax);
   pointCode1 = CompOutCode(x1,y1,xmin,ymin,xmax,ymax);
   do {
      if (pointCode0==0 && pointCode1==0) { /*prihvati duz i iz
         accept=true;
         done=true;
      }
      else if (pointCode0 & pointCode1) != 0)
         done=true; /* odbaci duz i izadji */
      else {        /* izaberi tacku koja je van pravougaonika
         if (pointCode0 != 0) then
            pointOut = pointCode0;
         else
            pointOut = pointCode1;
         if (pointOut & TOP) {
            x:=x0+(x1-x0)*(ymax-y0)/(y1-y0);
            y:=ymax;
         }
         else (pointOut & BOTTOM) {
            x:=x0+(x1-x0)*(ymin-y0)/(y1-y0);
            y:=ymin;
         }
         else if (pointOut & RIGHT) {
            y:=y0+(y1-y0)*(xmax-x0)/(x1-x0);
            x:=xmax;
         }
         else if (pointOut & LEFT) {
            y:=y0+(y1-y0)*(xmin-x0)/(x1-x0);
            x:=xmin;
         }
         if (pointOut==pointCode0) {
            x0=x; y0=y; pointCode0=CompOutCode(x0,y0);
         }
         else {
            x1=x; y1=y; pointCode1=CompOutCode(x1,y1);
         }
      }
   }
   while(!done);
   if (accept)
      MidpointLine(x0,y0,x1,y1)
}
```

### 5.2.3 Cyrus-Beckov algoritam

- Cyrus-Beckov algoritam primenljiv je na bilo kakav konveksni kliping region. Liang-Barsky varijanta optimizovana je za specijalni slučaj — pravougaonik sa horizontalnim/vertikalnim stranicama.

- Parametarski oblik duži $P_0P_1$:

$$P(t) = P_0 + (P_1 - P_0)t$$




- $N \cdot (P_1 - P_E) < 0$ za tačku $P_1$ koja je unutar poligona;

- $N \cdot (P_0 - P_E) > 0$ za tačku $P_0$ koja je izvan poligona;

- $N \cdot (P(t) - P_E) = 0$ za tačku $P_0$ koja je na stranici poligona.

- Neka je $D = \overrightarrow{P_0P_1}$

- uslov iz kojeg se može izračunati tačka preseka:

$$N \cdot (P(t) - P_E) = 0$$

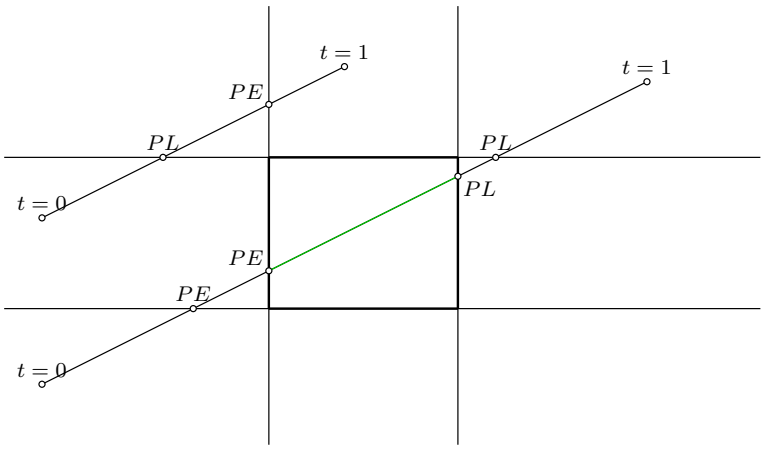
$$N \cdot (P_0 + (P_1 - P_0)t - P_E) = 0$$

$$N \cdot (P_0 - P_E + (P_1 - P_0)t) = 0$$

$$t = \frac{N \cdot (P_0 - P_E)}{-N \cdot (P_1 - P_0)}$$

$$t = \frac{N \cdot (P_0 - P_E)}{-N \cdot D}$$

- Proverava se da važi $D \neq 0$ (tačke $P_0$ i $P_1$ su različite), $N \neq 0$ (što ne važi samo u slučaju greške), $-N \cdot D \neq 0$ (stranica i duž koja se secka nisu paralelne).




- Potrebno je izvršiti klasifikaciju na „potencijalno ulazne" i „potencijalno izlazne" tačke.
- $N \cdot D < 0 \Rightarrow PE$ (ugao $> 90°$) (PE = potentially entering)
- $N \cdot D > 0 \Rightarrow PL$ (ugao $< 90°$) (PL = potentially leaving)
- Sve preseke je potrebno sortirati, izabrati maksimum od ulaznih i $t = 0$ i minimum od izlaznih i $t = 1$.
- $P_E$ u izračunavanjima može biti bilo koja tačka stranice kliping regiona, npr. njeno teme.
- Jednostavnosti radi, pretpostavlja se da duž $P_0P_1$ nije paralelna stranicama poligona. Ako je paralelna, vrši se posebna analiza.

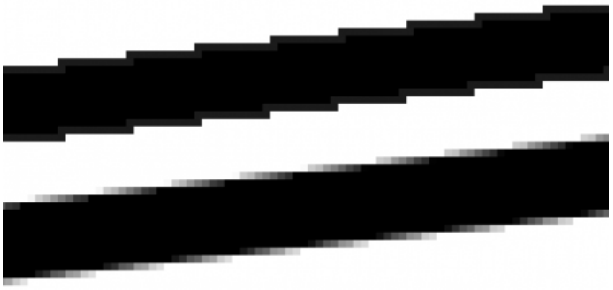
```
izracunaj Ni i odredi P_Ei za svaku stranicu;
za svaku duz koju treba iseckati uradi sledece:
  if (P1==P0)
     seckaj kao tacku
  else  {
     t_E=0; t_L=1;
     za svakog kandidata za presek uradi sledece:
        if N_i * D <> 0 then  {  /* ignorisi paralelne
           izracunaj t;
           upotrebi znak N_i * D da kategorizujes kao P
           if (PE) t_E = max(t_E,t);
           if (PL) t_L = min(t_L,t);
        }
        if (t_E>t_L)
           return nil;
        else
           return P(t_E) i P(t_L);
  }
```

- Liang-Barsky varijanta:

| Kliping ivica | normala $N$ | $P_E$ | $P_0 - P_E$ | $t = \frac{N \cdot (P_0 - P_E)}{-N \cdot D}$ |
|---|---|---|---|---|
| left: $x = x_{min}$ ... | $(-1, 0)$ | $(x_{min}, y_{min})$ | $(x_0 - x_{min}, y_0 - y_{min})$ | $\frac{-(x_0 - x_{min})}{x_1 - x_0}$ |

## 5.3 Antialiasing

- *Aliasing efekat* nastaje zbog toga što su objekti kao duži, poligoni, krugovi itd. neprekidni, dok je raster uređaj diskretan. Figura dobijena od orginalne figure diskretizacijom na rasterskom uređaju zove se alias.
- Aliasing se manifestuje na sledeće načine:
  - reckaste/stepenaste ivice
  - pogrešno prikazani fini detalji ili tekstura
  - mali objekti (npr. manji od jednog piksela) mogu da budu ignorisani ili da njihov uticaj bude preveliki
- *Antialiasing* su tehnike koje koriste različite intenzitete osvetljenosti za postizanje veće vizuelnu rezoluciju



Primer duži bez korišćenja antialiasing tehnika (gore) i sa korišćenjem antialiasing tehnika (dole)

- Postoje dva osnovna antialiasing metoda:
  - tretirati piksel ne kao tačku nego kao oblast.
  - povećati rezoluciju za koju se izvršava izračunavanje, a rezultat prikazati na postojećoj rezoluciji (nadsemplovanje)

### 5.3.1 Antialiasing — tretiranje piksela kao oblasti

- Tretirati piksel ne kao tačku (kao u osnovnim verzijama algoritama) nego kao oblast.
- Na primer, u algoritmu za popunjavanje poligona, izračunava se da li piksel (kao matematička tačka) pripada ili ne pripada unutrašnjosti poligona; ako pripada unutrašnjosti, cela oblast piksela se boji istom bojom i jedan deo te oblasti može da bude u spoljašnjosti idealnog poligona. Rezultat je karakteristična reckasta ivica poligona.

- Ako su raspoložive nijanse (nijanse sive ili boje) ovaj efekat je moguće ublažiti: umesto da se svaki piksel oboji datom bojom ili ne, treba izabrati pogodnu boju (koja odgovara fragmentu oblasti koja se nalazi unutar poligona).
- Na primer, ako je spoljašnjost poligona bela, a unutrašnjost i ivice treba obojiti crnom bojom, onda granične piksele treba obojiti bojom intenziteta proporcionalnog delu piksela (razmatranog kao kvadrat) koji pripada unutrašnjosti poligona.
- Bresenhamov algoritam je moguće jednostavno modifikovati tako da daje aproksimaciju dela (fragmenta) piksela koji se nalazi sa jedne strane prave.

### 5.3.2 Antialiasing — nadsemplovanje (supersampling)

- Povećati rezoluciju za koju se izvršava izračunavanje, a rezultat prikazati na postojećoj.
- Uređaji za prikaz imaju svoja praktična ograničenja, pa rezultat mora da se prilagodi takvom izlazu.
- Prikazivanje na izlazu sa manjom rezolucijom se realizuje tehnikom „uprosečavanja“ (eng. averaging).
- Postoje više vrsta uprosečavanja:
  - ravnomerno/uniformno (računa se prosek vrednosti elemenata matrice koja okružuje piksel). Naredne matrice se koriste za smanjivanje rezolucije 2 puta i 4 puta:

| | |
|---|---|
| 1 | 1 |
| 1 | 1 |

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |

  - sa težinskim faktorima (računa se zbir vrednosti pomnožene sa težinskim faktorima i onda se deli zbirom težinskih faktora). Matrica za smanjivanje rezolucije 3 puta (analogno matrica za smanjivanje rezolucije 5 puta):

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

  - boja se odredjuje na osnovu slučajno izabranih tačaka koje pripadaju oblasti piksela.

Glava 6

# Geometrijski algoritmi

Domen:

- kontinualni geometrijski problemi
- diskretni geometrijski problemi
- primeri: određivanje najbližih tačaka; najvećeg nagiba; konveksni omotač itd.

## 6.1 Ispitivanje da li tačka pripada unutrašnjosti poligona

- u opštem slučaju složenost $O(n)$
- za konveksne poligone može se primeniti efikasniji algoritam:
  - neka je niz temena poligona $p_1$, $p_2$, ..., $p_n$ i neka je $Q$ tačka za koju treba odrediti da li pripada unutrašnjosti poligona.
  - ako je $n = 3$, provera se izvršava neposredno
  - ako je $n > 3$, neka je $k = [n/2]$; ako je $Q$ sa iste strane prave $p_1p_k$ kao i tačka $p_2$, onda $Q$ pripada unutrašnjosti datog poligona ako i samo ako pripada unutrašnjosti poligona $p_1p_2 \dots p_k$ (voditi računa o specijalnom slučaju kada $Q$ pripada pravoj $p_1p_k$)
  - složenost $O(\log n)$

## 6.2 Određivanje prostog mnogougla

- **Problem:** dato je $n$ tačaka u ravni, takvih da nisu sve kolinearne. Povezati ih zatvorenom prostom poligonalnom linijom.
- `procedure ProstMnogougao(p_1, p_2, ... p_n: tacke u ravni);`

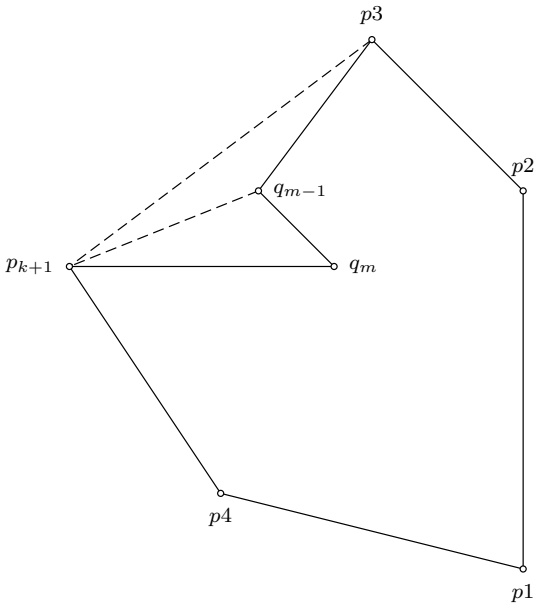
```
begin
   promeniti oznake tako da p1 bude tacka sa najvecom x
   koordinatom, a ako ima vise takvih, onda ona od njih
   koja ima najmanju y koordinatu;
```

```
   for i:=2 to n do
        izracunati ugao alpha_i izmedju prave p_1-p_i i x ose
   sortirati tacke prema uglovima alpha_i
   (ako za neki ugao ima vise tacaka, onda ih sortirati prema
   rastojanju od p_1);

   Trazeni mnogougao je odredjen dobijenom listom tacaka
end.
```

- složenost: $O(n \log n)$

## 6.3 Određivanje konveksnog omotača: Grahamov algoritam

- **Problem:** Za datih $n$ tačaka u ravni odrediti konveksni omotač.
- jedan algoritam: „uvijanje poklona“ (složenost $O(n^2)$)
- Osnovna hipoteza: ako je dato $n$ tačaka u ravni, uređenih na osnovu algoritma `prost_mnogougao`, onda umemo da konstruišemo podskup skupa prvih $k$ tačaka takav da odgovarajući konveksni omotač pokriva prvih $k$ tačaka.




- kada se doda jedna tačka, proverava se ugao koji zahvataju dve poslednje ivice; ako nova tačka pripada unutrašnjosti tekućeg poligona (tj. ako je ugao $q_{m-1} - q_m - p_{k+1}$ na slici manji od opružnog ugla), ona se dodaje nizu; inače se dodaje nizu ali se mora proveriti da li treba izbaciti neku od prethodnih tačaka.

```
procedure GrahamovAlgoritam(p_1, p_2, ... p_n: tacke u ravni);
```

```
begin
    promeniti oznake tako da p_1 bude tacka sa najvecom x
    koordinatom, a ako ima vise takvih, onda ona od njih
    koja ima najmanju y koordinatu;

    koristeci algoritam ProstMnogougao urediti tacke u odnosu na p_1;
    (neka je dobijen niz tacaka p_1, p_2, ... p_n);

    q_1 := p_1;
    q_2 := p_2;
    q_3 := p_3;
    m := 3;
    for k:=4 to n do
       while ugao izmedju q_{m-1}-q_m i q_m-p_k >=PI do
           m := m-1;
       m:=m+1;
       q_m := p_k;

   Trazeni mnogougao je odredjen dobijenom listom tacaka q_i
end.
```

- `PI` označava broj $\pi = 3.14...$
- složenost: $O(n \log n)$ (vraćacenjem unazad svaka tačka može biti obrisana iz niza najviše jednom!)

## 6.4 Određivanje najvećeg nagiba

- **Problem:** dato je $n$ tačaka u ravni; odrediti među njima dve takve da duž koja ih povezuje ima najveći nagib. Složenost algoritma treba da bude $O(n \log n)$
- **Rešenje:** najpre se tačke sortiraju prema $x$ koordinatama rastuće a onda se među nagibima duži koje povezuju *uzastopne* tačke bira najveći i on odgovara traženom paru.
- korektnost se zasniva na tvrđenju: ako između tačaka $A_0$ i $A_k$ u sortiranom redosledu postoje još neke tačke $A_1$, $A_2$, ..., $A_{k-1}$ onda postoje takve dve uzastopne tačke $A_i$ i $A_{i+1}$ takve da je nagib $A_iA_{i+1}$ veći ili jednak nagibu $A_0A_k$ (dokaz indukcijom).
- složenost: $O(n \log n)$

## 6.5 Određivanje maksimalnih tačaka

- **Problem:** Za tačku $P$ ravni kažemo da *dominira* tačkom $Q$ ako su i $x$ i $y$ koordinate tačke $P$ veće ili jednake od $x$ i $y$ koordinata tačke $Q$. Tačka $P$ je *maksimalna* u datom skupu tačaka $S$ ako nijedna od tačaka tog skupa ne dominira tačkvom $P$. Opisati algoritam reda $O(n \log n)$ za određivanje svih maksimalnih tačaka datog skupa $S$ od $n$ tačaka.

- **Rešenje:** Najpre sortiramo u opadajućem poretku sve tačke po vrednostima $x$ koordinata. Ako ima više tačaka sa istom $x$ koordinatom, onda uzimamo onu sa najvećom $y$ koordinatom, a ostale tačke zanemarujemo. Tačka $P_1$ je sigurno maksimalna. Neka je $Y$ vrednost njene $y$ koordinate. Tačka $P_2$ je maksimalna ako je vrednost $y$ koordinate veća od $Y$; u tom slučaju ažuriramo $Y$ (dobija vrednost $y$ koordinate tačke $P_2$) i dodajemo $P_2$ u skup maksimalnih tačaka. Inače, $P_2$ ne ulazi u skup maksimalnih tačaka, a postupak nastavljamo dalje analogno.

Glava 7

# Geometrijske osnove

## 7.1 Jednačine prave i ravni

- Jednačina prave u ravni je

$$ax + by + c = 0$$

- Jednačina ravni je

$$Ax + By + Cz + D = 0$$

  Normala ravni je $[A, B, C]$.

- Ako tri nekolinearne tačke $P_1$, $P_2$ i $P_3$ pripadaju ravni, onda je vektor $P_1P_2 \times P_1P_3$ kolinearan vektoru normale ravni $[A, B, C]$. Ako su tačke $P_1$, $P_2$, $P_3$ kolinearne, onda ovim tačkama nije određena ravan i vektorski proizvod $P_1P_2 \times P_1P_3$ jednak je $0$

- Rastojanje tačke $(x, y, z)$ od ravni određeno je jednakošću

$$d = \frac{Ax + By + Cz + D}{\sqrt{A^2 + B^2 + C^2}}$$

  Za sve tačke sa jedne strane ravni, vrednosti $Ax + By + Cz + D$ imaju isti znak (dakle, da bi se odredilo sa koje strane ravni je neka tačka dovoljno je izračunati vrednost $Ax + By + Cz + D$.

## 7.2 Vektori

Vektor je $n$-torka vrednosti. U daljem tekstu smatrajmo da su te vrednosti iz skupa realnih brojeva.

Vektori se sabiraju tako što se sabiranju vrednosti pojedinačnih elemenata, na primer:

$$v_1 + v_2 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 + x_2 \\ y_1 + y_2 \end{bmatrix}$$

Množenje vektora skalarom:

$$av = a \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax \\ ay \end{bmatrix}$$

Skalarni proizvod dva vektora:

$$v_1 \cdot v_2 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \cdot \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = x_1x_2 + y_1y_2$$

ili

$$v_2 \cdot v_2 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \cdot \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = x_1x_2 + y_1y_2 + z_1z_2$$

Vektori dimenzije 2, mogu da opisuju tačke u Dekartovoj ravni, a vektori dimenzije 3, mogu da opisuju tačke u Dekartovom prostoru.

Dužina vektora $v$ (označava se sa $|v|$) jednaka je $\sqrt{v \cdot v}$.

Jedinični vektor istog smera i pravca kao $v$ određen je sa $v/|v|$.

Ugao $\phi$ između dva vektora $v$ i $u$ može se izračunati na osnovu sledeće veze: $v \cdot u = |v||u|cos\phi$.

## 7.3 2D transformacije

**Translacija:**

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

u matričnoj formi:

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

tj.

$$P' = P + T$$

**Skaliranje:**

$$x' = s_x \cdot x \quad y' = s_y \cdot y$$

(nije ne nužno uniformno, tj. nije nužno $s_x = s_y$) u matričnoj formi:

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

tj.

$$P' = S \cdot P$$

**Rotacija** (rotacija oko koordinatnog početka; uglovi su pozitivno orijentisani):

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

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

u matričnoj formi:

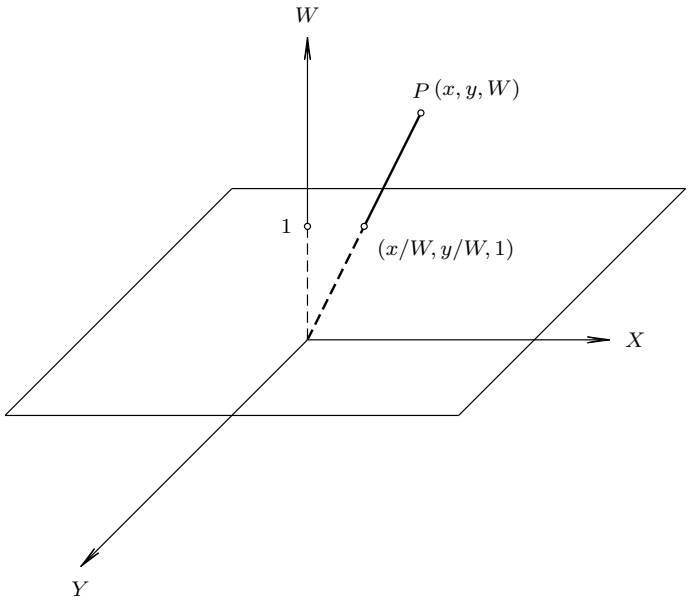
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

tj.

$$P' = R \cdot P$$

### 7.3.1 Homogene koordinate

- Bilo bi dobro da translacija ($P' = P + T$), skaliranje ($P' = S \cdot P$) i rotacija ($P' = R \cdot P$) imaju istu formu.
- Korsiteći homogene koordinate sve ove tri transformacije imaju formu množenja matrica.
- Homogene koordinate se koriste u geometriji od pedesetih godina, a u računarskoj grafici od šezdesetih godina dvadesetog veka.
- Homogenim koordinama 2D tačka se reprezentuje trojkom $(x, y, W)$.
- Dve trojke $(x, y, W)$ i $(x', y', W')$ reprezentuju istu tačku ako postoji broj $t$ takav da je $x = tx'$, $y = ty'$ i $W = tW'$.
- Bar jedna od vrednosti $x$, $y$, $W$ nije $0$.
- Ako je $W \neq 0$, onda tačku možemo reprezentovati u obliku $(x/W, y/W, 1)$ i $(x/W, y/W)$ zovemo dekartovskim koordinatama homogene tačke.
- Tačka za $(x, y, 0)$, je beskonačno daleka tačkama u pravcu $(x, y)$.
- Homogenim $2D$ tačkama odgovaraju trojke. Te trojke mogu da odgovaraju tačkama u Dekartovom prostoru.
- Skupu svih trojki kojima odgovara jedna homogena tačka odgovara prava Dekartovog prostora. Zaista, sve tačke oblika $(tx, ty, tW)$ ($t \neq 0$) pripadaju jednoj pravoj Dekartovog prostora.
- Ako *homogenizujemo* (podelimo sa $W$) jednu tačku, onda dobijamo njenu reprezentaciju $(x/W, y/W, 1)$ koja u dekartovskom smislu pripada ravni $W = 1$.



### 7.3.2 Transformacije u homogenim koordinatama

**Translacija:**

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

Kompozicija translacija je translacija:

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

**Skaliranje:**

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

Kompozicija skaliranja je skaliranje:

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

**Rotacija:**

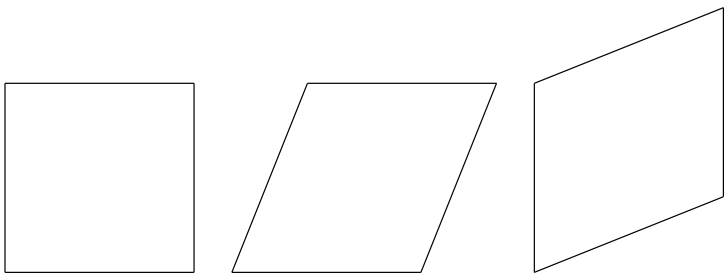
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Kompozicija rotacija je rotacija.

**Transformacija istezanja (*shear*):**

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

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



$SH_x : x' = x + ay$ $SH_y : y' = y + bx$

### 7.3.3 Izometrijske i afine transformacije

- Matrica je ortogonalna ako vektori koji čine njene kolone predstavljaju ortonormiranu bazu tj. ako je intenzitet svake kolone jednak 1 i ako je skalarni proizvod svake dve vrste jednak 0.
- Izometrijske transformacije čuvaju uglove i dužine (ponekad se zovu i *rigid-body* transformacije).
- Svakoj *izometrijskoj transformaciji* odgovara transformaciona matrica oblika:

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

  gde je njena gornja leva podmatrica $2 \times 2$ ortogonalna, tj. važi:

  - $a^2 + c^2 = 1$
  - $b^2 + d^2 = 1$
  - $ac + bd = 0$

- Važi i obratno, ovakvim matricama odgovaraju izometrijske transformacije.
- Rotacijama i translacijama odgovaraju ovakve matrice. Bilo kom nizu rotacija/translacija odgovara ovakva matrica.
- *Afine transformacije* čuvaju kolinearnost, proporcije i paralelnost, ali ne čuvaju uglove i dužine.
- Translacija, rotacija, refleksija, skaliranje i istezanje su afine transformacije i bilo koji niz rotacija/translacija/skaliranja/istezanja je afina transformacija.
- *Linearne transformacije* su rotacije, skaliranje i istezanje (i one se mogu opisati množenjem $2 \times 2$ matricama).
- Svaka afina transformacija je kompozicija jedne linearne transformacije i jedne translacije.

### 7.3.4 Kompozicije transformacija

- Transformacije izražene homogenim koordinatama jednostavno se kombinuju. To ilustruje sledeći primer.
- Rotacija oko tačke $P$ (koja nije koordinatni početak):
  - transliraj $P$ u koordinatni početak
  - rotiraj
  - transliraj koordinatni početak u $P$

$$T(x_1, y_1) \cdot R(\varphi) \cdot T(-x_1, -y_1) =$$

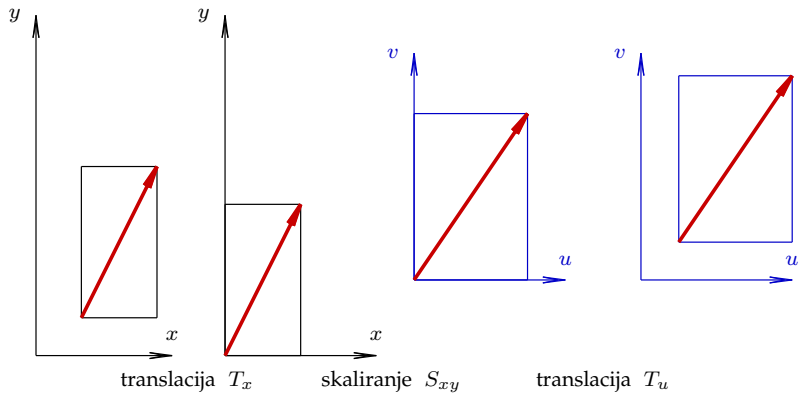
$$= \begin{bmatrix} 1 & 0 & x_1 \\ 0 & 1 & y_1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{bmatrix} =$$

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

- Analogno za skaliranje u odnosu na neku tačku.
- Generalno, transformacije ne komutiraju. U nekim slučajevima komutiraju, na primer, translacije međusobno komutiraju; rotacije oko iste tačke međusobno komutiraju i sl.

### 7.3.5 Preslikavanje slike u prozor

- Preslivanje figure iz jednog koordinatnog sistema u drugi je standardni problem, na primer, kod preslikavanja slike iz nekog koordinatnog sistema u neki prozor ekrana.
- Pretpostavimo da treba preslikati sadržaj pravougaonika sa levim donjim temenom $(x_{min}, y_{min})$ i gornjim temenom $(x_{max}, y_{max})$ u koordinatnom sistemu $(x, y)$ (pri čemu su stranice pravougaonika paralalelne osama) u pravougaonik sa levim donjim temenom $(u_{min}, v_{min})$ i gornjim temenom $(u_{max}, v_{max})$ u koordinatnom sistemu $(u, v)$ (pri čemu su stranice pravougaonika paralalelne osama) u pravougaonik.




- Svaku tačku pravougaonika treba
  - preslikati translacijom koja tačku $(x_{min}, y_{min})$ preslikava u koordinatni početak:

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

– preslikati skaliranjem koje tačku $(x_{max}, y_{max})$ preslikava u tačku $(u_{max}, v_{max})$;

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

– preslikati translacijom koja koordinatni početak preslikava u tačku $(u_{min}, v_{min})$:

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

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

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

### 7.3.6 Inverzne transformacije

- Inverznoj transformaciji za transformaciju kojoj odgovara matrica $M$ odgovara matrica $M^{-1}$.

- Inverzna transformacija translaciji $T$

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

je translacija kojoj odgovara matrica:

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

- Inverzna transformacija skaliranju $S$

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

je skaliranje kojem odgovara matrica:

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

### 7.3.7 Efikasnost izračunavanja

- Umesto da za svaku tačku primenjujemo množenje jednom matricom ili nizom matrica, obično najpre izračunamo matricu kompozicije i pokušavamo da iskoristimo njenu specifičnu formu.
- Često su transformacione matrice oblika:

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

pa je umesto množenja matrica dovoljno koristiti jednakosti:

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

$$y' = x \cdot c + y \cdot d + t_y$$

- Za transformacije složenih objekata u realnom vremenu mora se voditi računa o svakom množenju. Na primer, u jednakostima (za rotaciju):

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

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

$\cos\varphi$ se može aproksimirati vrednošću 1 ako je ugao $\varphi$ mali i time date jednakosti postaju:

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

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

(vrednost $\sin\varphi$ se, naravno, računa samo jednom a zatim koristi kao konstanta).

- Da bi se smanjilo nagomilavanje greške, bolje je u drugoj jednakosti koristiti $x'$ umesto $x$:

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

$$y' = x' \sin\varphi + y = (x - y \cdot \sin\varphi)\sin\varphi + y = x\sin\varphi + y(1 - \sin^2\varphi)$$

## 7.4 3D transformacije

- Homogene koordinate za prostor definišu se analogno homogenim 2D koordinatama:
  - Tačka $(x, y, z)$ 3D prostora reprezentuje se četvorkom $(x, y, z, W)$.
  - Dve četvorke reprezentuju istu tačku, ako i samo ako je jedna četvorka umnožak druge
  - Četvorka $(0, 0, 0, 0)$ nije dozvoljena
  - Standardna reprezentacija za tačku $(x, y, z, W)$ gde je $W \neq 0$ je $(x/W, y/W, z/W, 1)$ (a postupak svođenja na standardnu reprezentaziju zovemo *homogenizacija*)
  - Svakoj tački odgovara prava u 4-dimenzionom prostoru
- Transformacije su reprezentovane matricama 4x4.
- Orijentacija uglova je po dogovoru pozitivna.

### 7.4.1 Osnovne transformacije

**Translacija:**

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

Kompozicija dve translacije je translacija.

**Rotacija oko $z$ ose:**

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

Kompozicija dve rotacija oko $z$ ose je rotacija oko ose $z$.

**Rotacija oko $x$ ose:**

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

**Rotacija oko $y$ ose:**

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

Sve navedene matrice imaju inverzne matrice. Translacije i rotacije su izometrijske transformacije i njima odgovaraju matrice čije su gornje-leve $3 \times 3$ matrice ortogonalne.

**Skaliranje:**

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

Kompozicija dva skaliranja je skaliranje.

**Istezanje (*shear*):**

$$\begin{bmatrix} 1 & sh_{xy} & sh_{xz} & 0 \\ sh_{yx} & 1 & sh_{yz} & 0 \\ sh_{zx} & sh_{zy} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

### 7.4.2 Kompozicije transformacija

Svakoj kompoziciji rotacija, skaliranja, translacija i istezanja odgovara neka matrica oblika:

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

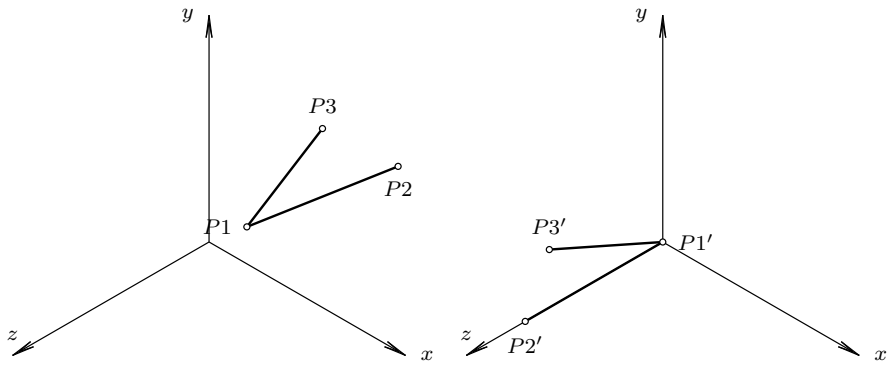
gde matrica $R^*$ opisuje kombinovani efekat rotacija, skaliranja i istezanja, a $t^*$ vektor koji opisuje kombinovani efekat svih translacija.

U skladu sa ovim, umesto korišćenja množenja matricom $4 \times 4$, može se koristiti efikasnije izračunavanje:

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

### 7.4.3 Primer kompozicije 3D transformacija

- Zadatak je odrediti transformaciju koja preslikava trojku tačaka $(P_1, P_2, P_3)$ u podudarnu trojku tačaka $(P_1', P_2', P_3')$.




- Jedan metod:
  - translirati $P_1$ u koordinatni početak;
  - rotirati oko $y$ ose tako da $P_1P_2$ pripada $(y, z)$ ravni;
  - rotirati oko $x$ ose tako da $P_1P_2$ pripada $z$ osi;
  - rotirati oko $z$ ose tako da $P_1P_3$ pripada $(y, z)$ ravni.
- Drugi metod se zasniva na neposrednom određivanju koeficijenata matrice transformacije (i uz korišénje svojstava ortogonalnih matrica).

### 7.4.4 Preslikavanje tačaka, pravih i ravni

- Matrice transformacija određuju slike pojedinačnih tačaka.
- Slika prave se može odrediti kao prava određena slikama dve tačke originalne prave.
- Slika ravni se može odrediti kao ravan određena slikama tri (nekolinearne) tačke originalne prave.
- Slika ravni se može odrediti i koristeći njen vektor normale.

### 7.4.5 Transformacije kao promena koordinatnog sistema

- Do sada su transformacije korišćene za preslikavanje skupa tačaka jednog koordinatnog sistema u isti taj koordinatni sistem. U tom kontekstu, možemo reći da je objekat tranformisan, a da je koordinatni sistem ostao isti. S druge strane, istu transformaciju možemo opisati 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, na primer, kada više objakata u svojim pojedinačnim koordinatnim sistemima treba objediniti u jedinstven koordinatni sistem.
- Za primenu koordinatnog sistema mogu se koristiti iste tehnike kao i za transformacije.
- Označimo sa $M_{i\leftarrow j}$ matricu transformacije koja prevodi koordinatni sistem $j$ u koordinatni sistem $i$. Neka je $P^{(i)}$ reprezentacija tačke $P$ u koordinatnom sistemu $i$, $P^{(j)}$ u sistemu $j$ i $P^{(k)}$ u sistemu $k$. tada važi

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

  i

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

- Navedena jednakost pokazuje i da je, za proizvoljnu promenu koordinatnog sistema, dovoljno imati opis svođenja iz različitih sistema na jedan sistem.
- Matrice $M_i$ imaju isti oblik kao i matrice za transformacije.
- Primer: sledeća matrica za promenu koordinatnog sistema:

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

  transformiše desno orijentisan sistem u levo orijentisan sistem (i obratno).

- Primer primene promene koordinatnog sistema: ako je automobil opisan u jednom koordinatnom sistemu i želimo da odredimo koordinate jedne tačke na točku dok se okreće, možemo najpre da za čitav točak promenimo koordinatni sistem (i izaberemo neki pogodan, npr. onaj u kojem je središte točka koordinatni početak i u kojem je osa točka jednaka $y$ osi) i u njemu računamo nove koordinate tačke.

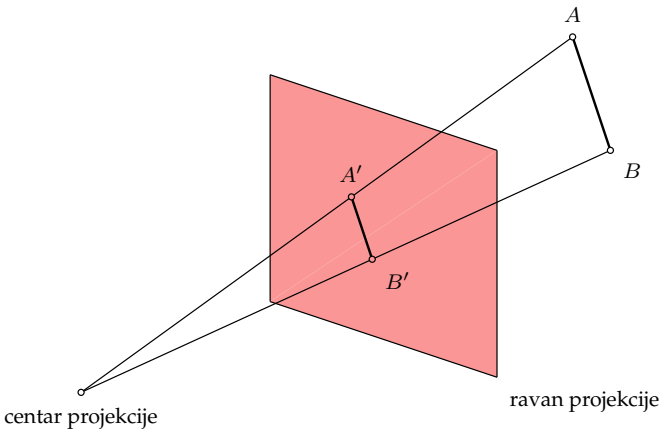
Glava 8

# Projektovanje

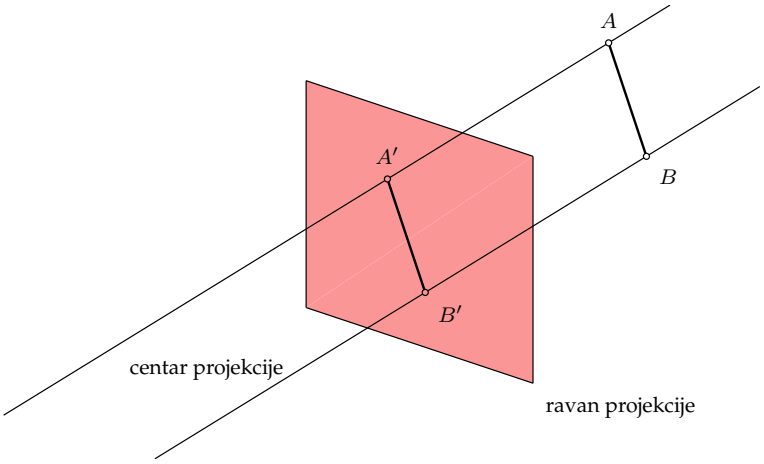
- Prikazivanje 3D objekata je znatno komplikovanije od prikazivanja 2D objekata.
- 3D objekti se obično prikazuju tako što se prikazuje njihova projekcija na ravan.
- Generalno: projekcija je preslikavanje iz koordinatnog sistema dimenzije $n$ u koordinatni sistem dimenzije manje od $n$.
- Od posebnog interesa za računarsku grafiku su projekcije iz 3D u 2D.
- Projekcije koje preslikavaju u ravan nazivamo *planarne projekcije*.
- Projekcija iz 3D u 2D je određena centrom projekcije i ravni projekcije.
- 3D objekti se prikazuju tako što se najpre odsecaju u odnosu na neku 3D oblast (3D view volume), a zatim projektuju na ravan.
- Sadržaj projekcije izabrane 3D oblasti na ravan projekcije naziva se *window* i on se transformiše u oblik za prikaz.

## 8.1 Tipovi projektovanja

- 3D planarne projekcije mogu biti:
  - Perspektivne (ako je centar projekcije na konačnom rastojanju od ravni projekcije).




– Paralelne (ako je centar projekcije na beskonačnom rastojanju od ravni projekcije).



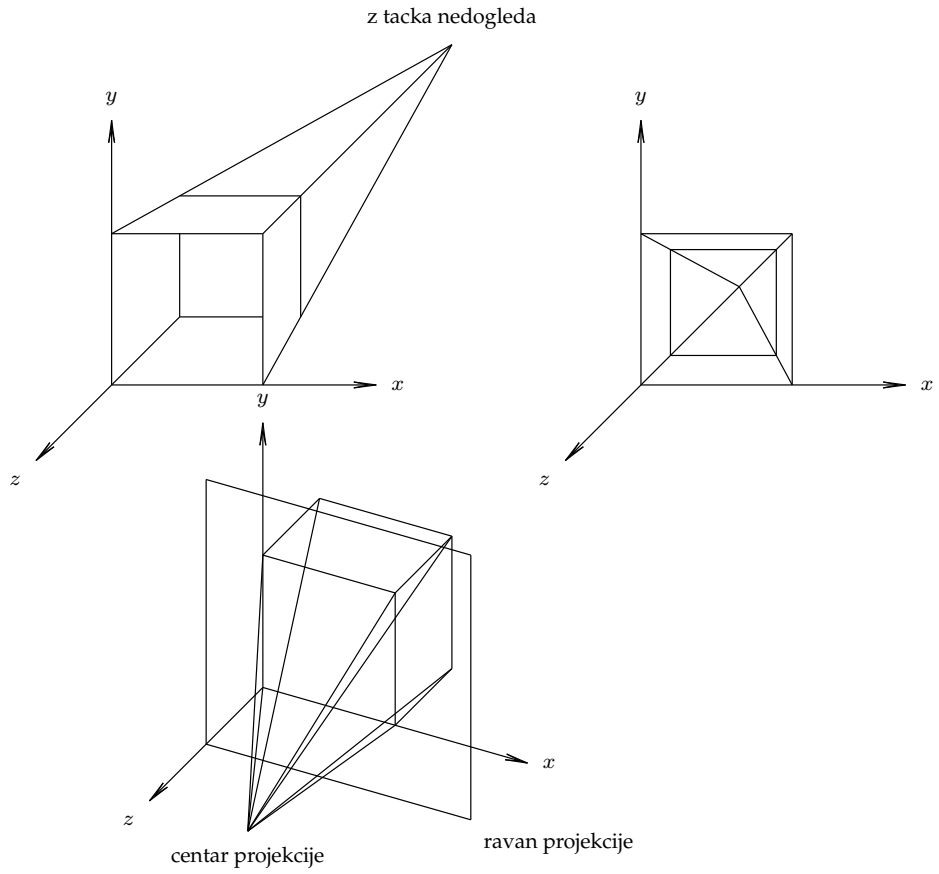

- Za perspektivnu projekciju eksplicitno se zadaje centar projekcije, a za paralelnu projekciju zadaje se pravac projekcije.
- Centar projekcije može biti reprezentovan homogenim koordinatama: $(x, y, z, 1)$.
- Pravac projekcije može biti zadat kao vektor, tj. kao razlika dve tačke (reprezentovane homogenim koordinatama): $(x, y, z, 1) - (x', y', z', 1) = (a, b, c, 0)$. Dakle, pravcu projekcije odgovara beskonačno daleka tačka. Perpektivna projekcija čiji je centar beskonačno daleka tačka je upravo paralelna projekcija.
- Vizualni efekat perspektivne projekcije odgovara ljudskom vizualnom sistemu i daje tzv. perspektivno skraćivanje. Mada slika u ovoj projekciji izgleda realistično, njen praktičan značaj može biti mali za određivanje tačnih oblika i dimenzija reprezentovanih objekata; paralelne prave se ne preslikavaju uvek u paralelne prave.
- Paralelna projekcija daje manje realističnu sliku; određivanje oblika i dimenzija je jednostavnije nego kod perspektivne projekcije; paralelne

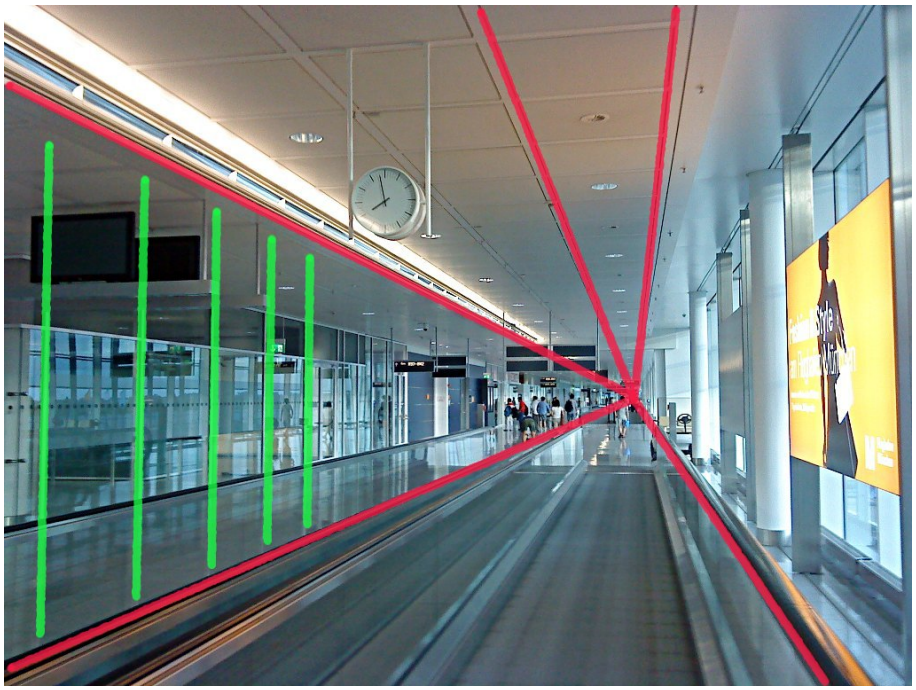
prave se preslikavaju u paralelne prave; kao i u perspektivnoj projekciji uglovi se čuvaju samo na ravnima koje su paralelne sa ravni projekcije.

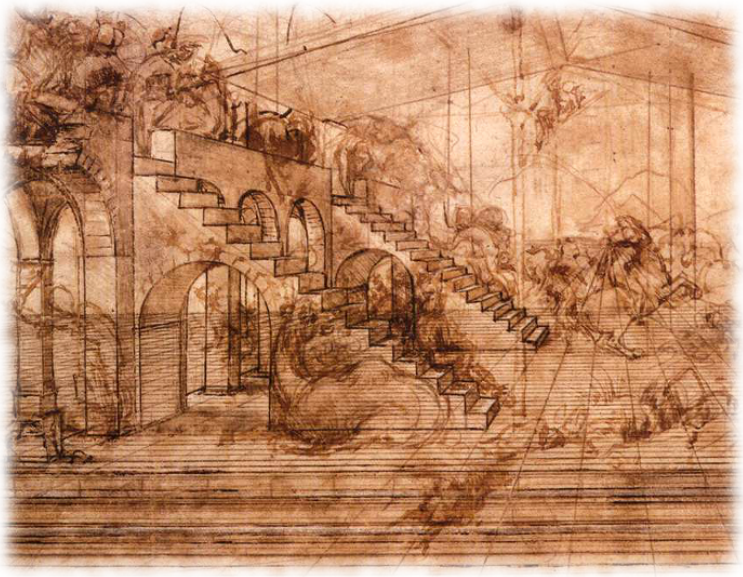
## 8.2 Perspektivna projekcija

- Perspektivne projekcije bilo kog skupa paralelnih pravih (koje nisu paralelne ravni projekcije) seku se u tzv. tački *nedogleda* (*vanishing point*).

- Ako je skup pravih paralelan sa koordinatnom osom, onda njenu tačku nedogleda zovemo *osna tačka nedogleda*. Ima najviše tri takve tačke i prema njihovom broju mogu se razvrstavati perspektivne projekcije. Naredne slike ilustruju perspektivne projekcije sa jednom takvom tačkom.







Leonardo da Vinči: studija za Poklonjenje mudraca (oko 1481)

## 8.3 Paralelna projekcija

- Paralelne projekcije se dele prema odnosu pravca projekcije i normale ravni projekcije. Ako su ovi pravci jednaki (ili suprotni), onda projekciju zovemo *ortogonalna*. Kod *kose projekcije* to nije slučaj.

- Najčešći tipovi ortogonalnih projekcija su *front-elevation*, *top-elevation* i *side-elevation* (u svakoj od ovih, ravan projekcije je normalna na jednu od koordinatnih osa).

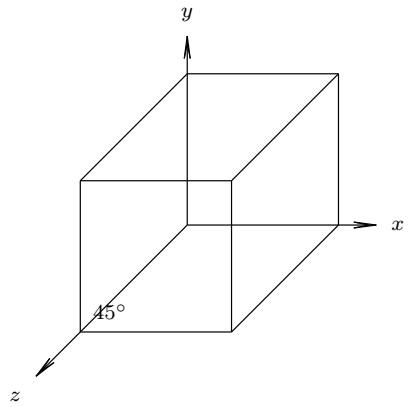
- Aksonometrijska projekcija koristi ravan projekcije koja nije paralalena nijednoj od koordinatnih osa (za razliku od perspektivnog projektovanja, skraćivanje je uniformno, i na njega ne utiče rastojanje od centra projekcije).

- Izometrijsko projektovanje je aksonometrijsko projektovanje u kojem normala ravni projektovanja zahvata podudarne uglove sa sve tri koordinatne ose (ovim svim jednakim dužinama odgovaraju jednake dužine projekcija).

- Kosa projekcija koristi ravan projekcije koja je normalna na neku od koordinatnih osa, a pravac projekcije nije jednak (niti suprotan) normali ravni projekcije.

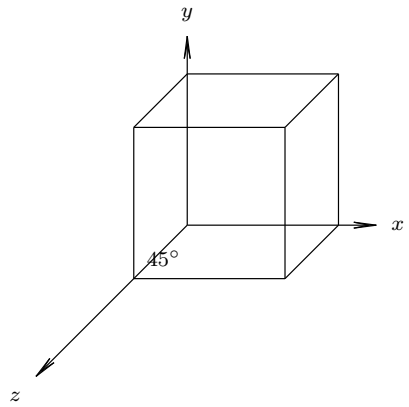
- Najčešće korišćeni tipovi kosih projekcija su *cavalier* i *cabinet*.

- *cavalier* projektovanje koristi pravac projektovanja koji zahvata ugao $45°$ sa normalom ravni projektovanja. Kao rezultat, projekcije duži normalnih na ravan projekcije imaju iste dužine kao i same duži (tj. za njih nema skraćivanja)



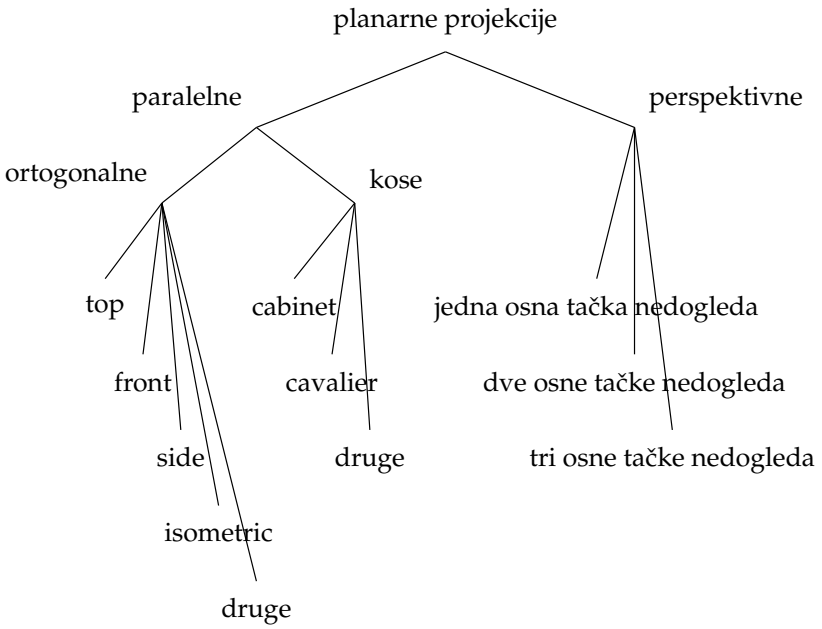

- Ove projekcije se međusobno mogu razlikovati po uglu koji zahvata pravac projekcije sa koordinatnim osama. Tim uglom je određen i ugao između projekcija pravih koje su paralelne sa ravni projekcije i koje su normalne na nju. Na gornjoj slici taj ugao je $45°$. Taj ugao je obično $45°$ ili $30°$

- *cabinet* projektovanje koristi pravac projektovanja koji zahvata ugao $arctg(2) \approx 63°$ sa normalom ravni projektovanja. Kao rezultat, projekcije duži normalnih na ravan projekcije imaju dva puta manje dužine u odnosu na same duži (što daje nešto realističniji prikaz)



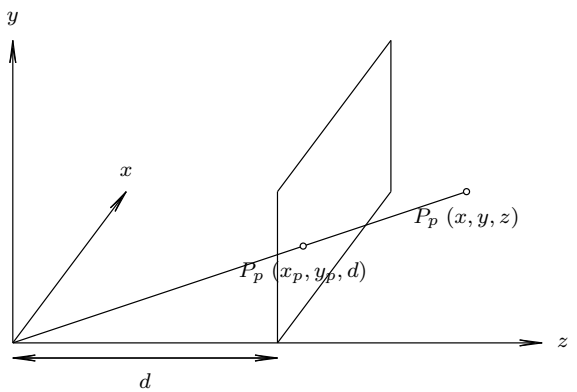

- Slično kao i u *cavalier* projektovanju, ove projekcije se međusobno mogu razlikovati po uglu koji zahvata pravac projekcije sa koordinatnim osama. Tim uglom je određen i ugao između projekcija pravih koje su paralelne sa ravni projekcije i koje su normalne na nju. Na gornjoj slici taj ugao je $45°$. Taj ugao je obično $45°$ ili $30°$.

## 8.4 Tipovi planarnih projekcija




## 8.5 Primer izračunavanja projekcija tačaka

- Razmotrimo sledeći jednostavan primer perspektivnog projektovanja takvog da je:
  - centar projektovanja koordinatni početak;
  - ravan projekcije je ravan $z = d$.



- Na osnovu sličnosti, važi

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

odakle je

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

(vrednosti nisu definisane za $z = 0$)

- Ovoj transformaciji odgovara matrica:

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

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

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

- U 3D dekartovskom sistemu $\left(\frac{x}{z/d}, \frac{y}{z/d}, d\right)$ je projekcija tačke $(x, y, z)$.

## 8.6 Opisivanje i imeplementiranje 3D → 2D projekcija

Ovo poglavlje govori o specifikovanju i detaljima neophodnim za implementiranje 3D → 2D projekcija. Ovaj postupak se može ilustrovati sledećom shemom:

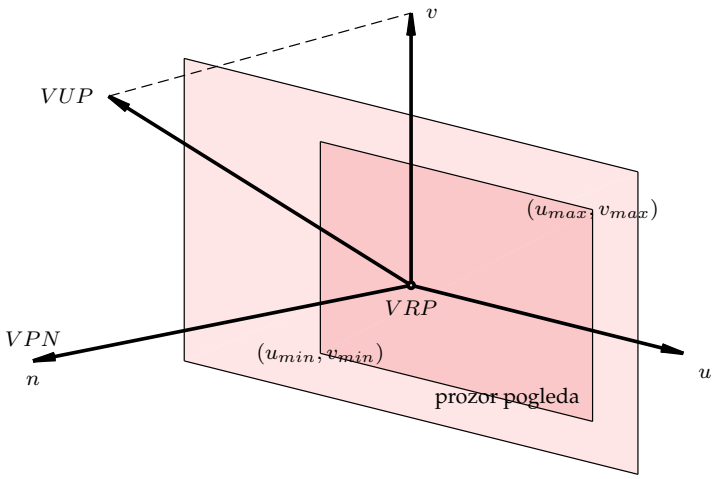
Objekti opisani u 3D "koordinatama sveta" → Seckanje u odnosu na zapreminu pogleda → Projektovanje na projekcionu ravan → Transformiranje u koordinate 2D uređaja → Objekti opisani u 2D koordinatama uređaja

Osnovni cilj je definisanje transformacije $N_{par}$ (za paralelne projekcije) i $N_{persp}$ (za perspektivne projekcije) koje preslikavaju tačku iz 3D koordinata iz oblasti posmatranja u normalizovane koordinate projekcije u 2D ravni.

U daljem tekstu ćemo projekcionu ravan zvati i *view plane* ili VP, oblast posmatranja *view volume* ili VV, ravan normale na projekcionu ravan *view plane normal* ili VPN, i referentnu tačku na projekcionoj ravni *view reference point* ili VRP. Naglasimo da projekciona ravan može biti iza, ispred ili može seći objekte iz oblasti posmatranja.

### 8.6.1 Referentni koordinatni sistem pogleda

Koordinatne ose $u$ i $v$ ravni VP deo su *referentnog koordinatnog sistema pogleda* (*viewing reference coordinate*, VRC, slika 8.1). VRC je zadat parametrima: ravan VP, tačka VRP, vektori VPN i VUP i oni određuju njegove ose $u$, $v$ (istovremeno i ose koordinatnog sistema u VP) i $n$. Koordinatni početak sistema VRC je tačka VRP. Vektor VPN određuje osu $n$ sistema VRC. Osa $v$ određena je projekcijom vektora VUP na VP. Time je jednoznačno određena i osa $u$, normalna na $n$ i $v$, tako da sistem bude orijentisan u skladu sa pravilom desne ruke.




Slika 8.1: Referentni koordinatni sistem pogleda

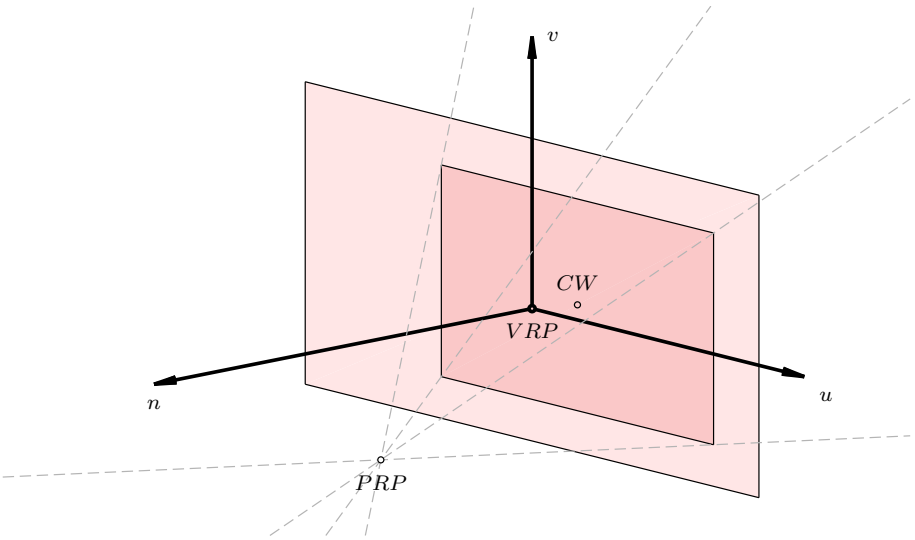
Deo ravni pogleda koji sadrži traženu projekciju (*prozor pogleda*, *view window*) je određen tačkama $(u_{min}, v_{min})$, $(u_{max}, v_{max})$ (slika 8.1). Središte ovog pravougaonika označavamo sa CW (center of projection window).

*Centar projektovanja* ili *referentna tačka projektovanja* (*projection reference point*) se obeležava sa PRP (slika 8.2).

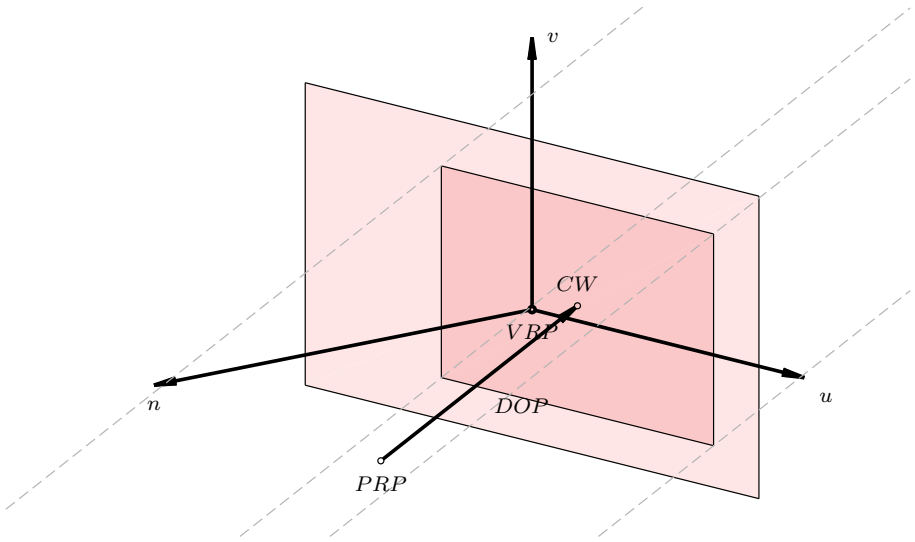
*Pravac projektovanja* (*direction of projection*), DOP, kod paralelnog projektovanja određen je tačkama PRP i CW (slika 8.3).

### 8.6.2 Specifikovanje zapremine pogleda

*Zapremina pogleda* (*view volume*) je inicijalno određena tačkom PRP i prozorom pogleda. Ona je inicijalno beskonačna: „beskonačna piramida“ za perspektivno projektovanje i „beskonačan paralelopiped“ za paralelno projektovanje. Obično se, međutim, ograničava oblast koja se projektuje. Ona se ograničava označenim vrednostima $F$ i $B$ koje određuju *prednju* i *zadnju ravan odsecanja*. Prednja i zadnja ravan odsecanja paralelne su ravni projektovanja i vrednosti $F$ i $B$ su njihova rastojanja od ravni projektovanja. Zapremine pogleda za perspektivno i paralelno projektovanje prikazane su na slikama 8.4 i 8.5.



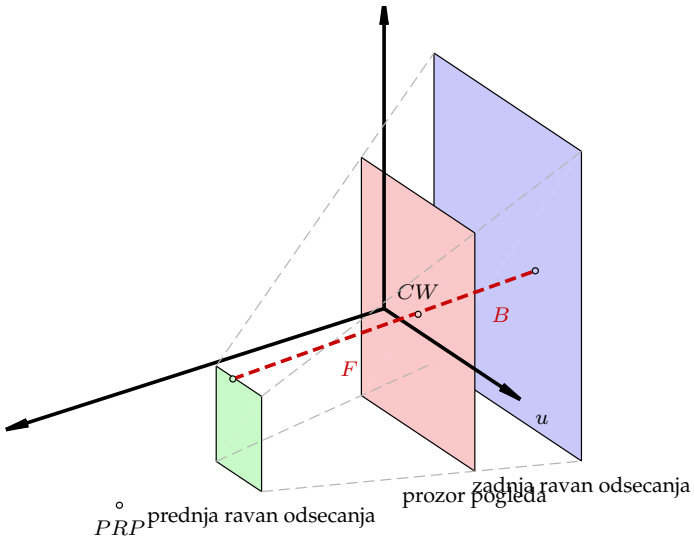

Slika 8.2: VRC i perspektivno projektovanje



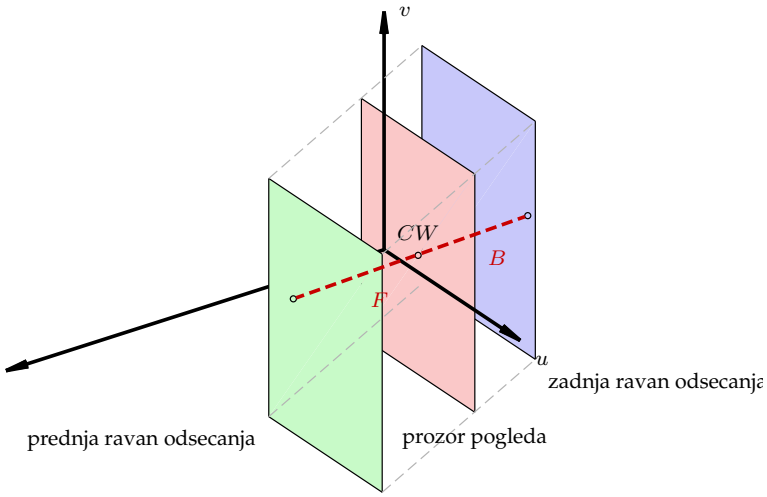

Slika 8.3: VRC i paralelno projektovanje

Sadržaj zapremine pogleda se preslikava u kvadar koji je poravnat sa osama VRC sistema, dajući kanonsku zapreminu pogleda ili *3D viewport* (slika 8.6). Kod paralelnog projektovanja, da bi bio prikazan žičani prikaz sadržaja 3D viewporta, dovoljno je zanemariti $z$ koordinatu. Za prikaz koji uzima u obzir vidljivost i sl., mora se uzimati u obzir $z$ koordinata.

Osnovni cilj u izračunavanju projekcije je određivanje matrice transformacija $N_{par}$ (za paralelno projektovanje) i $N_{per}$ (za perspektivno projektovanje)

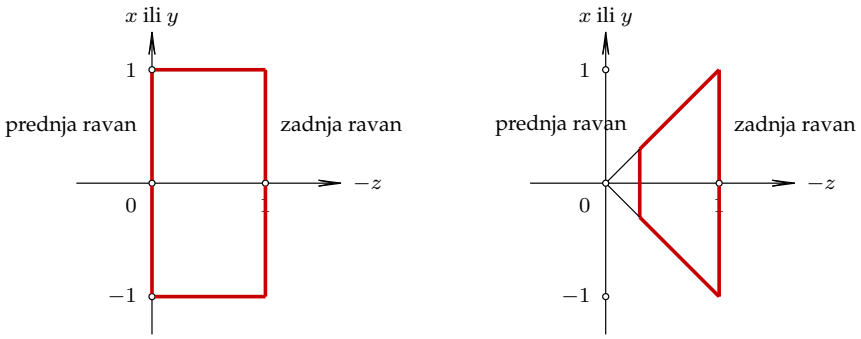



Slika 8.4: Zapremina pogleda za perspektivno projektovanje




Slika 8.5: Zapremina pogleda za paralelno projektovanje

koja transformiše tačke zadate u koordinatama sveta u tačke zadate normalizovanim koordinatama. Na njih se primenjuje kliping algoritam i jednostavno 3D $\rightarrow$ 2D projektovanje.




Slika 8.6: Kanonska zapremina pogleda za paralelno (levo) i perspektivno (desno) projektovanje

### 8.6.3 Implementiranje paralelnog projektovanja

U ovom slučaju kanonska zapremina pogleda (kreirana preslikavanjem u normalizovane koordinate) je kvadar određen ravnima: $x = -1, x = 1, y = -1, y = 1, z = 0$ i $z = -1$.

Koraci potrebni za preslikavanje u kanonsku zapreminu pogleda su:

1. Translirati VRP u koordinatni početak.

$$T(-VRP) = \begin{bmatrix} 1 & 0 & 0 & -vrp_x \\ 0 & 1 & 0 & -vrp_y \\ 0 & 0 & 1 & -vrp_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2. Rotirati VRC tako da je poravnat sa svetskim koordinatnim sistemom. Neka je

$$R_z = \frac{VPN}{|VPN|} \tag{8.1}$$

$$R_x = \frac{VUP \times R_z}{|VUP \times R_z|} \tag{8.2}$$

$$R_y = R_z \times R_x \tag{8.3}$$

Može se pokazati da je matrica tražene transformacije jednaka:

$$R = \begin{bmatrix} r_{1,x} & r_{2,x} & r_{3,x} & 0 \\ r_{1,y} & r_{2,y} & r_{3,y} & 0 \\ r_{1,z} & r_{2,z} & r_{3,z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

gde vrste odgovaraju vektorima $R_x$, $R_y$, $R_z$.

3. Primeniti istezanje (po $x$ i $y$ osi) tako da je pravac projektovanja poravnat sa $z$ osom.

   Vektor projektovanja DOP može se izračunati na sledeći način (($dop_x, dop_y, dop_z$) su koordinate vektora DOP u svetskom koordinatnom sistemu, a $(prp_u, prp_v, prp_n)$ su koordinate tačke PRP u referentnom sistemu pogleda; ova dva koordinatna sistema su nakon prethodnog koraka usaglašena):

$$DOP = \begin{bmatrix} dop_x \\ dop_y \\ dop_z \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{u_{max}+u_{min}}{2} - prp_u \\ \frac{v_{max}+v_{min}}{2} - prp_v \\ -prp_n \\ 0 \end{bmatrix}$$

   pa je matrica istezanja jednaka

$$H_{par} = \begin{bmatrix} 1 & 0 & -\frac{dop_x}{dop_z} & 0 \\ 0 & 1 & -\frac{dop_y}{dop_z} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4. Transformisati u kanonske koordinate i translirati i skalirati u kanonsku zapreminu pogleda (kvadar).

   Translacija centra prednje strane zapremine odsecanja u koordinatni početak:

$$T_{par} = \begin{bmatrix} 1 & 0 & 0 & -\frac{u_{max}+u_{min}}{2} \\ 0 & 1 & 0 & -\frac{v_{max}+v_{min}}{2} \\ 0 & 0 & 1 & -F \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

   Skaliranje:

$$S_{par} = \begin{bmatrix} \frac{2}{u_{max}-u_{min}} & 0 & 0 & 0 \\ 0 & \frac{2}{v_{max}-v_{min}} & 0 & 0 \\ 0 & 0 & \frac{1}{F-B} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Dakle, tražena matrica projektovanja je

$$N_{par} = S_{par} T_{par} H_{par} RT(-VPR)$$

### 8.6.4 Implementiranje perspektivnog projektovanja

U ovom slučaju kanonska zapremina pogleda (kreirana preslikavanjem u normalizovane koordinate) je zarubljena piramida koja pripada kvadru određenom ravnima: $x = -1, x = 1, y = -1, y = 1, z = 0$ i $z = -1$.

Koraci potrebni za preslikavanje u kanonsku zapreminu pogleda su:

1. Translirati VRP u koordinatni početak — kao kod paralelnog projektovanja.

2. Rotirati VRC tako da je poravnat sa svetskim koordinatnim sistemom. — kao kod paralelnog projektovanja.

3. Translirati centar projektovanja (tačku PRP) u koordinatni početak.

$$T(-PRP) = \begin{bmatrix} 1 & 0 & 0 & -prp_u \\ 0 & 1 & 0 & -prp_v \\ 0 & 0 & 1 & -prp_n \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4. Primeniti istezanje (po $x$ i $y$ osi) tako da je prava PRP-CW bude poravnata sa $z$ osom — analogno kao korak 3 kod paralelnog projektovanja.

5. Skalirati u kanonsku zapreminu pogleda (zarubljenu piramidu).

$$S_{persp} = \begin{bmatrix} \frac{2prp_n}{(u_{max}-u_{min})(prp_n-B)} & 0 & 0 & 0 \\ 0 & \frac{2prp_n}{(v_{max}-v_{min})(prp_n-B)} & 0 & 0 \\ 0 & 0 & \frac{1}{prp_n-B} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Dakle, tražena matrica projektovanja je

$$N_{persp} = S_{persp}H_{par}T(-PRP)RT(-VPR)$$

Primetimo da matrice $N_{par}$ i $N_{persp}$ ne utiču na homogenu koordinatu $w$ (jer su kombinacije translacija, rotacija, skaliranja i istezanja), pa nije neophodno normalizovati koordinate po $w$ da bi se prešlo na dekartovske 3D koordinate.

Često je pogodno primeniti dodatni korak u izračunavanju matrice $N_{persp}$ — korak koji transformiše kanonsku zarubljenu piramidu u kanonski kvadar (isti kao kod paralelnog projektovanja), tako da se u nastavku isti algoritmi mogu primenjivati i za paralelno i za perspektivno projektovanje. Matrica za ovu transformaciju je:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{1+z_{min}} & \frac{-z_{min}}{1+z_{min}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

gde je $z_{min} = (prp_n - F)/(B - prp_n)$. Korišćenjem matrice $M$ možemo dobiti matricu za novu transformaciju koja preslikava objekte u kvadar kao zapreminu pogleda: $N^*_{persp} = MN_{persp}$.

### 8.6.5 Seckanje

Seckanje objekata tako da preostanu samo oni delovi koji pripadaju zapremini pogleda vrši se primenom 3D verzije Cohen-Sutherlandovog ili Cyrus-Beckovog algoritma.

### 8.6.6 Projektovanje

Kada su svi objekti transformisani u kanonsku zapreminu pogleda i nakon što je urađeno seckanje, preostaje samo da se izvrši jednostavno projektovanje. Za paralelno projektovanje:

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

Za perspektivno projektovanje (ako nije zarubljena piramida transformirasana u kvadar kao kanonsku zapreminu pogleda):

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

(na osnovu primera urađenog u poglavlju 8.5, sa vrednošću $d = -1$).

### 8.6.7 Transformisanje u 3D viewport

Kanonska zapremina pogleda može da se transformiše u tzv. 3D viewport odakle će se dalje jednostavno dobiti vrednosti za 2D prikaz. Neka su $(x_{vmin}, y_{vmin}, z_{vmin})$ donji levi ugao i $(x_{vmax}, y_{vmax}, z_{vmax})$ gornji desni ugao 3D viewport-a.

Translirati donji levi ugao kanonske zapremine pogleda u koordinatni početak, skalirati u dimenzije 3D viewport-a i, na kraju, translirati koordinatni početak u donji levi ugao 3D viewport-a:

$$\begin{bmatrix} 1 & 0 & 0 & x_{vmin} \\ 0 & 1 & 0 & y_{vmin} \\ 0 & 0 & 0 & z_{vmin} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{x_{vmax}-x_{vmin}}{2} & 0 & 0 & 0 \\ 0 & \frac{y_{vmax}-y_{vmin}}{2} & 0 & 0 \\ 0 & 0 & \frac{z_{vmax}-z_{vmin}}{1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

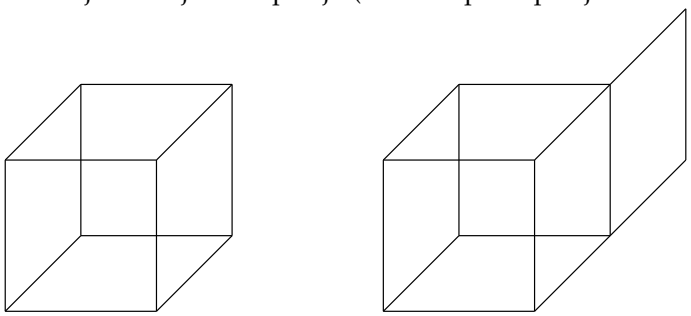
Za konačni 2D prikaz treba homogenizovati koordinate tačke i ignorisati $z$ koordinatu.

Glava 9

# Modelovanje tela

- Krivama u 2D i površima u 3D mogu se opisati oblasti i tela; međutim, nije uvek lako proveriti da li skup krivih u 2D određuje neku oblast ili ne i da li skup površi u 3D određuje neko telo.
- U mnogim aplikacijama je važno opisati neko telo (ili neku oblast u 2D), ispitivati da li je neka tačka u unutrašnjosti tela ili nije, da li dva tela imaju presek i da li se uklapaju na potreban način (npr. komponente robota i sl.) Dobar opis objekata treba da omogući i automatsko generisanje instrukcija za mašinu koja može da napravi takav objekat. Dobar opis objekata omogućava i određivanje svojstva svetlosti koja se odbija ili prolazi kroz telo.
- Sve pomenute aplikacije (i ne samo one) su primeri *modelovanja tela*. Potreba da se objekti opisuju kao tela rezultovalo je razvojem mnoštva pristupa za reprezentovanje tela.

Opisivanje tela na osnovu površi često nije pogodno. Za dati skup ivica ili površi nije uvek jasno koje telo opisuju (i da li uopšte opisuju neko telo):



## 9.1 Uslovi za modelovanje tela

Reprezentacija tela treba da zadovoljava sledeće uslove (u što je moguće većoj meri):

- *Domen* reprezentacije treba da bude dovoljno velik da može da pokrije koristan/potreban skup fizičkih objekata

- Reprezentacija treba da bude *jednoznačna, nedvosmislena* ili *potpuna*; to znači da jedna reprezentacija treba da opisuje jedno i samo jedno telo.

- Reprezentacija je *jedinstvena* ako bilo koje dato telo može da opiše na tačno jedan način (ovo svojstvo olakšava ispitivanje da li su dva objekata jednaka).

- *Precizna* reprezentacija opisuje tela bez aproksimiranja.

- Reprezentacija treba da bude takva da je nemoguće napraviti reprezentaciju koja ne opisuje nijedno telo.

- Reprezentacija treba da bude takva da je lako napraviti reprezentaciju velikog broja tela.

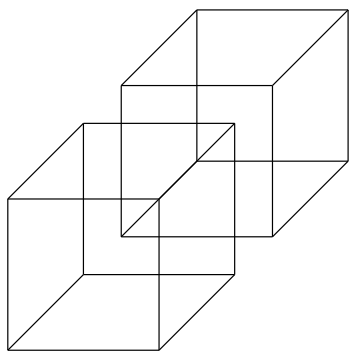
- Reprezentacija treba da bude zatvorena za rotaciju, translaciju i druge transformacije (tj. primenjivanje ovih transformacija na ispravna tela daje ponovo ispravna tela (u zadatoj reprezentaciji)).

- Reprezentacija treba da bude *kompaktna* radi čuvanja prostora.

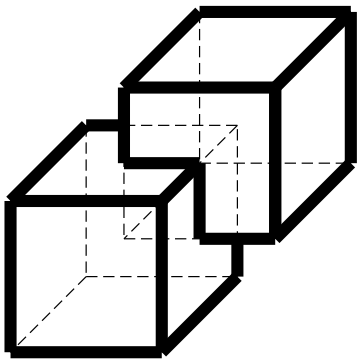
- Reprezentacija treba da je takva da omogućava efikasne algoritme.

## 9.2 Bulovske skupovne operacije

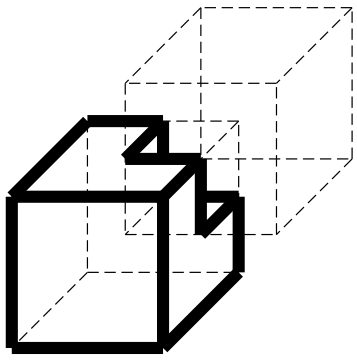
- Jedan od najjednostavnijih i najintuitivnijih metoda za kombinovanje tela su bulovske skupovne operacije, kao što su unija, presek, razlika.



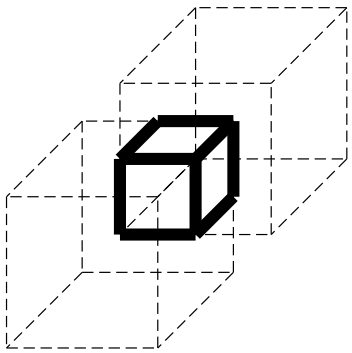
objekti $A$ i $B$



$A \cup B$



$A \setminus B$



$A \cap B$

- Kako bi se obezbedilo da rezultat operacije nad dva tela daje ili neko telo ili prazan skup, umesto običnih bulovskih operacija koriste se *ispravljene* (eng. *regularized*) bulovske operacije

- Ispravljenu bulovska operacija $op$ označavamo sa $op^*$ i definišemo na sledeći način:

$$A \ op^* \ B = \text{zatvorenje(unutrašnjost}(A \ op \ B))$$

## 9.3 Mreža poligona

- Mreža poligona (eng. mash) je skup povezanih poligonskih površi (obično trouglova).

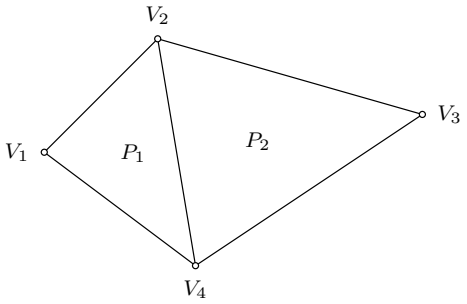
- Mreža poligona može reprezentovati bilo koje poliedar, ali može da aproksimira druge tipove tela.

- Postoji više načina za reprezentovanje mreže poligona.

### 9.3.1 Reprezentacije mreže poligona

- **eksplicitna reprezentacija:** svaki poligon je opisan nizom svojih temena; za jedan poligon, ovo rešenje je prostorno-efikasno; međutim, za više poligona ovo rešenje može da troši mnogo prostora jer veliki broj temena može da pripada raznim poligonima iz skupa. Dodatno, veliki problem u ovoj reprezentaciji je i to što ne postoji eksplicitna informacija o zajedničkim temenima i ivicama. Da bi se proverilo da li je jedno teme istovremeno 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 svaka ivica će biti (nepotrebno) iscrtavana dva puta.

  **reprezentacija sa pokazivačima na liste indeksa temena:** svaka tačka (svako teme) je zapisano tačno jednom u listi temena. Poligon je reprezentovan kao lista indeksa temena. Ova reprezentacija štedi prostor. Dodatno, ona omogućava lako ispitivanje da li dva poligona imaju zajedničku tačku. S druge strane, i u ovoj reprezentaciji se svaka ivica crta dva puta.

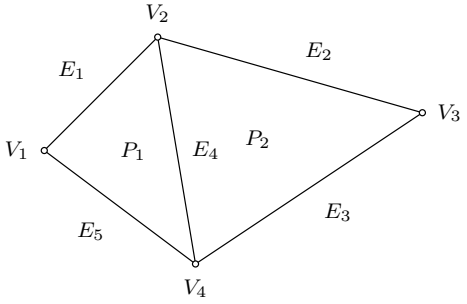



$V = (V_1, V_2, V_3, V_4) = ((x_1, y_1, z_1), ..$

$P_1 = (1, 2, 4)$

$P_2 = (4, 2, 3)$

  **reprezentacija sa pokazivačima na liste indeksa ivica:** Alternativno, poligoni mogu biti reprezentovani kao liste indeksa ivica, dok je svaka ivica opisana parom tačaka. Dodatno, u opisu ivice mogu da budu i poligoni kojima pripada (radi efikasnijih obrada).




$V = (V_1, V_2, V_3, V_4) = ((x_1, y_1, z_1), ..$

$E_1 = (V_1, V_2, P_1, null)$

$E_2 = (V_2, V_3, P_2, null)$

$E_3 = (V_3, V_4, P_2, null)$

$E_4 = (V_4, V_2, P_1, P2)$

$E_5 = (V_4, V_1, P_1, null)$

$P_1 = (E_1, E_4, E_5)$

$P_2 = (E_2, E_3, E_4)$

- svaka od navedenih reprezentacija ima prednosti i mane i izbor treba napraviti u skladu sa konkretnim zadatkom

### 9.3.2 Konzistentnost mreže poligona

- Mreže poligona se često definišu interaktivno ili kombinovanjem reprezentacija više objekata. Zbog toga je potrebno proveravati da li su zadržana očekivana svojstva, kao što su:
  - svaka ivica se koristi bar jednom (a najviše $n$ puta, gde $n$ u nekim primenama može da bude i veće od 2)
  - svako teme se koristi u bar dve ivice
  - mreža poligona je povezana
  - mreža poligona je nultog roda (nema „rupa“)
  - ...
- Ovakvi i slični uslovi definišu *konzistentnost mreže poligona* (i potrebno je da ona može jednostavno i brzo da se proverava)

### 9.3.3 Reprezentacija ruba

- Rubovi ili granice tela se često definišu mrežama poligona i sl.
- Potrebno je, za svaku posebnu primenu, izabrati pogodan način za reprezentovanje mreže poligona
- Za proveravanje da li je telo dobro definisano može se koristiti Ojlerova formula (za poliedarske površi):
  - za površi nultog roda:

    $$V - E + F = 2$$

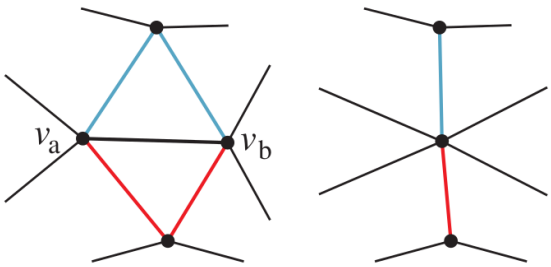
    gde je $V$ broj temena, $E$ broj ivica, $F$ broj pljosni.
  - za površi sa „rupama“:

    $$V - E + F - H = 2(C - G)$$

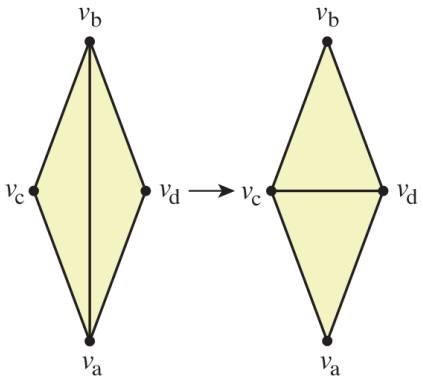
    (gde je $H$ broj „rupa“ na stranama poliedarske površi, $G$ broj „rupa“ koje prolaze kroz objekt i $C$ je broj zasebnih delova tela)

### 9.3.4 Primeri operacije nad mrežom poligona

- Edge-collapse (uključuje izbor tačke koja će zameniti stranicu koja se eliminiše):



- Edge-swap (proizvode manje neželjenih vizuelnih efekata trouglovi koji nemaju visok *aspect ratio*):



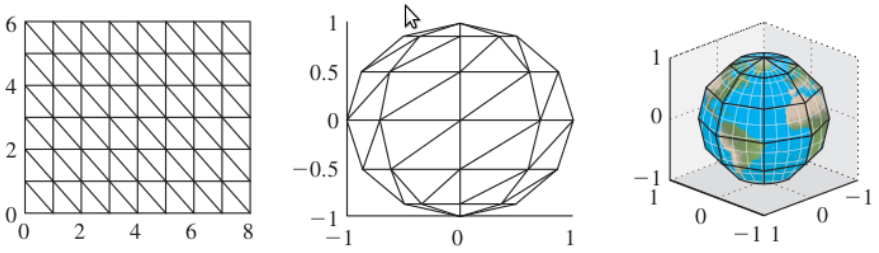

### 9.3.5 Primer funkcije nad mrežama poligona — teksture

Temena poligona tačkama u teksturi pridružuje osoba koja pravi model, ali u nekim situacijima to može biti urađeno lako ili čak automatski. Razmotrimo mrežu trouglova $n \times k$ gde je $n = 6$ i $k = 8$ (slika dole levo). Svako teme $(i, j)$ ove mreže možemo preslikati u 3D prostor na sledeći način:

$\theta = 2\pi j/(k-1)$
$\phi = \pi/2 + \pi i/(n-1)$
$X = \cos(\theta)\cos(\phi)$
$Y = \sin(\phi)$
$Z = \sin(\theta)\cos(\phi)$

Dobijena figura u prostoru prikazana je na slici dole u sredini.

Ako imamo i $100 \times 200$ sliku Zemlje za teksturu, možemo pridružiti koordinate teksture $100i/(n1)$, $200j/(k1)$ temenu mreže $(i, j)$, i rezultat je figura prikazana na slici dole desno.




## 9.4 Instanciranje primitivama

- U instanciranju primitivama, sistem za modelovanje definiše skup primitiva (tj. skup osnovnih 3D tela) koji je relevantan za konkretnu primenu (npr. primitiva može biti pravilna piramida sa promenljivim brojem stranica).

- Parametrizovane primitive mogu se shvatiti kao familije delova čiji članovi se razlikuju samo po nekoliko parametara (što je važan koncept u CAD (computer-aided design), poznat kao *tehnologija grupe*)
- Primitivno instanciranje se često koristi i za relativno složene objekte (kao npr. zupčanici) koje bi bilo teško opisati u terminima bulovskih operacija.
- Iako se može graditi hijerarhija primitivnih instanci, svaki list je i dalje poseban objekt. U instanciranju primitivama nema ograničenja poput npr. uslova da se kombinovanjem primitivnih objekata bulovskim operacijama dobijaju objekti višeg reda.

## 9.5 Reprezentacija kretanjem (sweep representations)

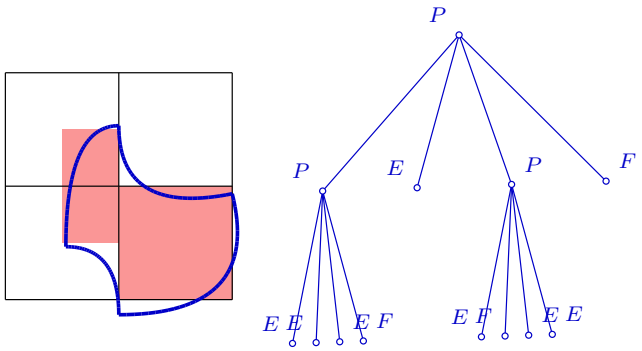
- Kretanje nekog objekta duž neke putanje u prostoru određuje telo koje zovemo *sweep*.
- Najjednostavnija tela ove vrste su definisana 2D oblašću koja se kreću duž prave; takva tela zovemo translacionim.
- Objekat koji se kreće ne mora da bude 2D.
- Proširenja ove reprezentacija dopuštaju da se objekat koji se kreće skalira tokom kretanja (i slična uopštenja).

## 9.6 Reprezentacije zasnovane na particionisanju prostora

- U reprezentacijama zasnovanim na particionisanju prostora (*spatial-partitioning representations*) telo se dekomponuje u skup susednih, nepresecajućih tela koja su obično jednostavnija od originalnog tela. Ta primitivna, osnovna tela mogu se razlikovati u veličini, obliku, orijentaciji i sl. Neki od primera ovog tipa reprezentacije su:
  - *dekomponovanje na ćelije*: ćelije su osnovni skup objekata (obično parametrizovanih) i njihovim kombinovanjem („lepljenjem“) mogu se dobiti nove ćelije
  - *enumeracija prostornog zauzeća* (eng. spatial-occupancy enumeration) je specijalni slučaj dekomponovanja na ćelije, pri čemu ćelije čine fiksiranu, pravilnu mrežu. Ove ćelije se često zovu *voxeli* (eng. *voxels, volume elements*), po analogiji sa pikselima.

    U ovom pristupu postoji problem sa prostorom, jer količina informacija je reda $O(n)$, gde je $n$ broj voksela duž svake ose. Prostorna zahtevnost može biti smanjena korišćenjem jedinica *oktri* (eng. octree) kod kojih se primenjuje binarna podela u stilu podeli-i-vladaj algoritama. U opštem slučaju, broj čvorova oktrija (koji odgovara potrebnom prostoru) je srazmeran površini objekta (koja je reda $O(n^2)$, a ne reda $O(n^3)$). Postoje efikasni algoritmi za obradu oktrija.

    Umesto da opisujemo oktri, opisaćemo jednostavniji, 2D analogon oktrija — *kuodtri* (eng. quadtree). U kuodtriju, ćelije su kvadrati, a




Slika 9.1: Ilustracija kuodtrija nivoa tri

svaki kvadrat je podeljen na četiri manja kvadrata (i tako na svakom nivou stabla). Slika 9.1 prikazuje kuodtree nivoa 3 koji opisuje planarni objekat iscrtan plavom linijom a aproksimiran obojenom površinom.

Čvor u kvodtriju može biti delimično pokriven objektom (P), potpuno pokriven (F) ili nepokriven (E). Listovi uvek imaju vrednost E ili F. Ova vrsta modela se naziva i model zasnovan na stablima.

Glava 10

# Opisivanje krivih i površi u 3D

- Glatke krive i površi su neophodne u mnogim primenama računarske grafike
- Modelovanje objekata može biti zasnovano na matematičkim karaktersitikama, aproksimacijama, interaktivnom definisanju itd.
- Za reprezentovanje 3D objekata često se koriste sledeći modeli:
  - mreža poligona
  - parametarske polinomske krive
  - parametarske binarne polinomske površi
  - kvadratne površi zadate implicitno
  - itd.

## 10.1 Parametarske kubne krive

- Poligonske linije i poligonske površi su prvog stepena i oni su deo po deo linearna aproksimacija krivih i površi. Da bi se postigla potrebna preciznost, u mnogim slučajevima je potrebno koristiti ogroman broj linija ili poligona.
- Često je pogodnije koristiti polignoske krive ili površi višeg reda. Poligonska aproksimacija je pogodna zbog jednostavne manipulacije, a kao veoma pogodan stepen često se uzima vrednost 3.

  Za parametarske krive, 3 je najmanji stepen takav da kriva ne mora da pripada ravni. Zaista, kriva stepena 2 je potpuno određena trima tačkama, a te tri tačke određuju ravan (kojoj pripada ta kriva).
- Aproksimativne krive mogu se opisati na jedan od načina:

  **eksplicitna reprezentacija:** $y = f(x), z = g(x)$ (problem: za jednu vrednost $x$ postoji samo jedna tačka na krivoj, te je u nekim slučajevima potrebno kombinovati više krivih)

**implicitna reprezentacija:** $f(x, y, z)$ (problem: može da ima više rešenja nego što nam treba; na primer, za modeliranje kruga možemo da koristimo $x^2 + y^2 = 1$, ali kako modelirati polukrug? Potrebno je koristiti dodatne uslove, ali to stvara druge probleme.

**parametarska reprezentacija:** $x = x(t), y = y(t), z = z(t),\ 0 \leq t \leq 1$. Ova reprezentacija nema većinu problema koje imaju prethodne dve. Za funkcije $x$, $y$ i $z$ se često koriste kubni polinomi. Time se zadata kriva aproksimira deo po deo polinomijalnim krivim.

- Kubni polinom koji opisuje deo krive $Q(t) = [x(t)\ y(t)\ z(t)]$ je oblika:

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x,$$

$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y,$$

$$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z,$$

$$0 \leq t \leq 1$$

- Ove jednakosti se mogu napisati i u formi:

$$Q(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} =$$

$$\begin{bmatrix} a_x & b_x & c_x & d_x \\ a_y & b_y & c_y & d_y \\ a_z & b_z & c_z & d_z \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} = C \cdot T$$

- Izvod krive $Q(t)$ je parametarski vektor tangente krive. Važi:

$$\frac{d}{dt}Q(t) = [\frac{d}{dt}x(t)\ \frac{d}{dt}y(t)\ \frac{d}{dt}z(t)] =$$

$$= \frac{d}{dt}T \cdot C = [3t^2\ 2t\ 1\ 0] \cdot C =$$

$$= [3a_x t^2 + 2b_x t + c_x \quad 3a_y t^2 + 2b_y t + c_y \quad 3a_z t^2 + 2b_z t + c_z]$$

- Da bi koeficijenti krive bili određeni dovoljno je imati njene vrednosti u četiri tačke, tj. dovoljno je imati (tri puta) četiri jednačine (svaka od funkcija $x, y, z$ ima po četiri koeficijenta, a vrednosti u iste četiri tačke mogu biti korišćene za određivanje sve tri funkcije)

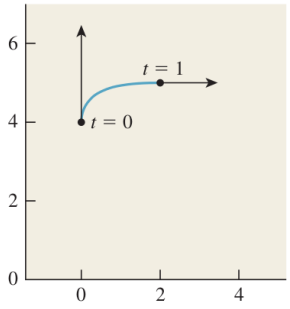
- Za zadavanje jednačina za određivanje krivih mogu se uzeti vrednosti krive u krajnjim tačkama, kao i vrednosti izvoda u krajnjim tačkama. Ovim vrednostima (tj. ovim skupom $3 \times 4$ jednačina) je kriva određena .

### 10.1.1 Nadovezivanje i tipovi neprekidnosti

- Za nadovezivanje krivih jednu na drugu možemo se rukovoditi sledećim kriterijumima:
  - ograničenja nad krajnjim tačkama;
  - ograničenja nad tangentnim vektorima (tj. izvodima);
  - ograničenja vezana za neprekidnost u krajnjim tačkama.
- Kažemo da kriva pripada klasi $C^0$ ako je neprekidna. Kažemo da kriva pripada klasi $C^1$ ako je diferencijabilna i njen izvod je neprekidan (i kažemo da je neprekidno diferencijabilna). Kažemo da kriva pripada klasi $C^2$ ako je diferencijabilna i njeni prvi i drugi izvod su neprekidni. Analogno se definiše klasa $C^n$.

### 10.1.2 Primer

- **Primer 1:** Neka se automobil 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 i kako skreće i usporava tako da je u trenutku $t = 1$ u poziciji (2,5) sa vektorom brzine (2,0) (videti sliku dole).




- Problem se može uopštiti na sledeći način. Neka su date pozicije $P$ i $Q$ i vektori brzine $v$ i $w$. 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 ima sledeći oblik:

  $\gamma(t) = (2t^3 - 3t^2 + 1)P + (-2t^3 + 3t^2)Q + (t^3 - 2t^2 + t)v + (t^3 - t^2)w =$
  $(1-t)^2(2t+1)P + t^2(-2t+3)Q + t(t-1)^2v + t^2(t-1)w.$

- **Primer 2:** Vektor tangente (tj. izvod od) $Q(t)$ je brzina tačke na krivoj u odnosu na parametar $t$. Slično, drugi izvod od $Q(t)$ je ubrzanje. Ako se duž parametarske kubne krive kreće kamera i u jednakim vremenskim intervalima pravi po jednu sliku, izvod od $Q(t)$ daje brzinu kretanja kamere, a drugi izvod od $Q(t)$ daje njeno ubrzanje. Da bi filmska sekvenca bila vizuelno prihvatljiva potrebno je da su i brzina i ubrzanje neprekidne na zajedničkim tačkama segmenata krive.

### 10.1.3 Tipovi krivih

Za različite namene, različiti tipovi krivih (trećeg stepena) mogu biti najpogoniji. Najčešće se koriste:

**Hermitove krive** (određuju se vrednostima u krajnjim tačkama i izvodima u krajnjima tačkama);

**Bezijerove krive** (zadaje se krajnjima tačkama i dvema dodatnim („kontrolnim“) tačkama koje ne moraju nužno da pripadaju krivoj (ali implicitno određuju izvode u krajnjim tačkama);

**B-splajnovi** određuje se na osnovu četiri *kontrolne tačke* (sa $C^2$ neprekidnošću).

### 10.1.4 Crtanje krivih

**Prvi pristup:** vrednost $t$ 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 i da je njene izvode lako opisati)

**Drugi pristup:** rekurzivna podela krive na segmente dok se ne dođe do pravih linija (tj. približno pravih linija); implementira se različito za različite tipove krivih, a razlikuju se i testovi da li se segment može aproksimirati pravom linijom.

## 10.2 Parametarske bikubne površi

- Analogno kao kubne krive.
- $Q(s, t)$ je površ trećeg reda, nad parametrima $s$ i $t$
- Tipovi:
  - Hermitove površi
  - Bezijerove površi
  - B-splajn površi
  - ...
- Prikazivanje bikubnih površi:
  - iterativnim izračunavanjem tačaka za konstantne promene parametara;
  - rekurzivnim podelama do „približno ravnih“ delova površi.

Glava 11

# Vidljivost (Rendering)

- Problem: za dati skup 3D objekata potrebno je odrediti koje ivice i koje površi su vidljive, ili iz centra projekcije (za perspektivno projektovanje) ili duž pravca projekcije (za paralelno projektovanje).
- Ovaj proces je poznat kao *određivanje vidljivih linija/površi* ili kao *eliminacija skrivenih linija/površi*.
- Problem je jednostavno formulisati, ali efikasna rešenja mogu da budu veoma komplikovana.
- Mnogi potproblemi (kao svetlost, boja) i tzv. antialiasing (npr. kako obojiti tačku koja je na granici između dva vidljiva objekta)

## 11.1 Vidljivost — dva opšta pristupa

**Prvi pristup** za svaki piksel slike određuje koji od $n$ objekata je vidljiv.

- Za svaki piksel slike uraditi:
  - Odrediti objekat najbliži posmatraču (duž prave određene projekcijom).
  - Obojiti piksel odgovarajućom bojom.
- Ovaj pristup se obično zove *image-precision*.
- Najpre je korišćen u ranim sistemima sa rasterskom grafikom (koji mogu da adresiraju relativno mali broj tačaka, ali mogu da prikazuju veliki broj objekata).

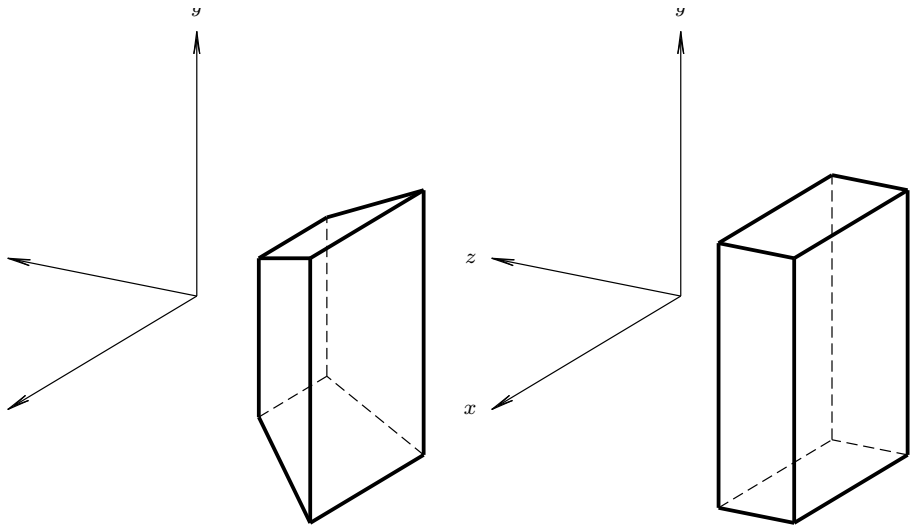
**Drugi pristup** je da se objekti porede međusobno direktno i da se eliminišu čitavi objekti ili njihovi delovi koji nisu vidljivi.

- za svaki objekat uraditi:
  - odrediti delove objekta koji nisu zaklonjeni drugim delovima ili drugim objektima
  - oboji taj deo odgovarajućom bojom
- ovaj pristup se obično zove *object-precision*

- najpre je korišćen u sistemima sa vektorskom grafikom (koji mogu da adresiraju ogroman broj tačaka, ali ne mogu da prikazuju veliki broj objekata).
- većina efikasnih algoritama zasnovana je na kombinaciji ova dva pristupa

## 11.2 Odsecanje i transformisanje zapremine pogleda

- Zadaju se prednja i zadnja ravan odsecanja (*front clipping plane* i *back clipping plane*.
- Ako je u pitanju paralelno projektovanje, testiranje da li tačka $P_1(x_1, y_1, z_1)$ zaklanja tačku $P_2(x_2, y_2, z_2)$ svodi se na test da li je $x_1 = x_2$ i $y_1 = y_2$; ako je u pitanju centralno projektovanje, onda je taj test znatno složeniji (zahteva deljenje) i zato se obično kod centralnog projektovanja najpre izvrši perspektivna transformacija koja jedno preslikavanja svodi na drugo kako bi složeni test bio zamenjen jednostavnim




zapremina pogleda pre i posle perspektivne transformacije

- Matrica transformacije:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{1+z_{min}} & \frac{-z_{min}}{1+z_{min}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

## 11.3 Algoritmi za određivanje vidljivosti

- Algoritmi za određivanje vidljivih linija:
  - Robert's algorithm (Roberts 1963)
  - Appel's algorithm (Appel 1967)

- z-buffer algorithm (Catmull 1974)
- algoritmi sa listama prioriteta
  - depth-sort algorithm (Newell, Newell, Sancha 1972)
  - binary space-partitioning trees (Fuchs, Kedem, Naylor 1980)
- scan-line algoritmi
- algoritmi zasnovani na podeli prostora (area-subdivision algorithms)
  - Warnock's algorithm (Warnock 1969)
  - Weiler-Atherton algorithm (Weiler, Atherton 1977)
  - subpixel area-subdivision algorithms
- algoritmi za krive površi
  - visible surface ray tracing
  - antialiased ray tracing

## 11.4 z-bafer algoritam

- Autor: Catmull 1974.
- Pripada grupi image-precision algoritama.
- Jednostavno se implementira i softverski i hardverski.
- Obično se koristi za real-time renderovanje.
- Koristi se u OpenGL-u.
- Zahteva postojanje memorijskog prostora ne samo za vrednost boje za svaki piksel (frejm bafer F), već i memorijskog prostora za z-vrednosti (tj. z-koordinate) za svaki piksel (z-bafer).
- Sve vrednosti u F baferu su incijalizovane na boju pozadine, a sve vrednosti u Z baferu su inicijalizovane na 0 (kao na z-koordinatu zadnje ravni odsecanja). Maksimalna moguća vrednost u z-baferu je z-koordinata prednje ravni odsecanja. (Analogno bi bilo ako se podrazumeva da je zadnja ravan odsecanja ravan $z = -1$).
- Na sve poligone (nije bitan poredak) koji reprezentuju scenu primenjuje se scan-conversion algoritam. Prilikom scan-konverzije za tačku $(x, y)$, ako ta tačka nije od posmatrača dalja od tačke koja je trenutno u baferima, onda vrednosti te tačke zamenjuju postojeće vrednosti u baferima.

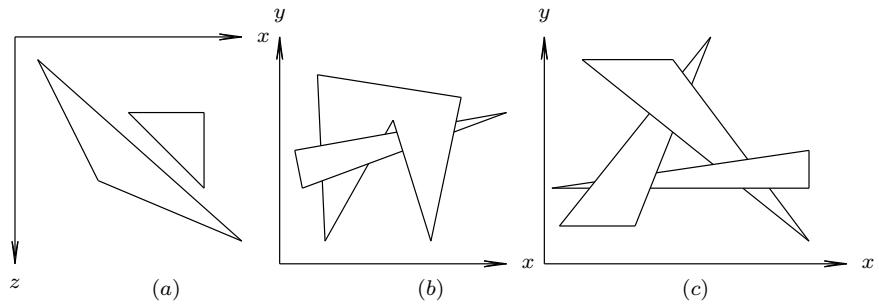
```
Procedure zBuffer
var
   pz: integer;
begin
 for y:=0 to YMAX do
  for x:=0 to XMAX do
   begin
    WritePixel(x,y,background_value);
```

```
    WriteZ(x,y,0)
   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.
```

- Nije potrebno sortiranje objekata pre primene algoritma
- Poligoni se pojavljuju na slici redom kojim se obrađuju
- Moguće su optimizacije računanja (izbegavanje množenja i sl. na bazi prirode scan-conversion algoritma)
- z-bafer algoritam ne zahteva nužno da primitivne površi budu poligoni
- z-bafer podaci (zajedno za oba bafera) mogu da budu čuvani zajedno sa generisanom slikom i da se toj sceni *naknadno* doda novi objekat.

## 11.5 Algoritam sortiranje dubine (Depth-sort algorithm)

- Autori: Newel, Newel, Sancha, 1972.
- Pripada grupi object-precision algoritama
- Najčešće se koristi za real-time renderovanje
- Osnovna ideja je da boji poligone u opadajućem poretku po rastojanju od posmatrača.
- Osnovna tri koraka:
  - sortirati sve poligone prema njihovoj najmanjoj (najdaljoj) z-koordinati
  - razreši sve moguće dileme u slučajevima kada se z-koordinate poligona međusobno preklapaju, deleći ih na manje poligone ako je to potrebno
  - primeni scan-conversion algoritam na sortirani niz poligona u opadajućem poretku (od daljih ka bližim).
- Pojednostavljena verzija ovog algoritma očekuje kao ulaz i eksplicitno zadate prioritete koji određuju poredak poligona. Ova pojednostavljena verzija algoritma se naziva *slikarev algoritam* (jer asocira na to kako se slikaju bliži predmeti preko daljih). U opštem slučaju ne garantuje ispravno renderovanje.




- Poligoni se obrađuju jedan po jedan. Za tekući poligon $P$ (najdalji od posmatrača) mora da se ispita svaki poligon $Q$ za koji se $z$ koordinate preklapaju sa $z$ koordinatama poligona $P$. Za $P$ i $Q$ se primenjuje sledeća lista testova:

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

- Ako bar jedan od testova uspe, onda se poligon $P$ isrtava scan-conversion algoritmom i sledeći poligon dobija ulogu poligona $P$.

- Ako nijedan od ovih testova ne uspe, onda smatramo da se poligoni $P$ i $Q$ potencijalno preklapaju, te proveravamo da li se poligon $Q$ može iscrtati pre poligona $P$. Testove 1, 2 i 5 ne treba ponavljati, a testove 3 i 4 treba ponoviti sa zamenjenim ulogama poligona $P$ i $Q$:

  3′ da li je čitav $Q$ sa suprotne strane ravni poligona $P$ u odnosu na posmatrača?
  4′ da li je čitav $P$ sa iste strane ravni poligona $Q$ kao i posmatrač?

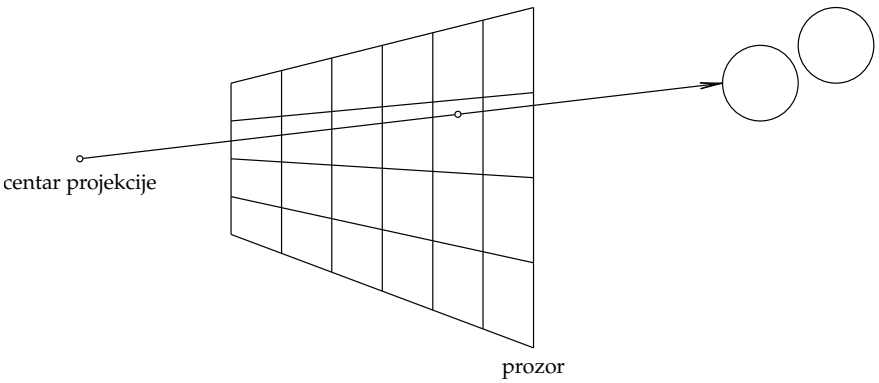
- Ako jedan od ovih testova uspe, onda poligoni $Q$ i $P$ zamenjuju uloge.

- U primeru $(b)$ ni ovi testovi ne omogućavaju razrešenje i onda ili $P$ ili $Q$ mora biti podeljen na poligone od strane ravni koja sadrži drugi polazni poligon (primenom kliping algoritma)

- Primer $(c)$ ilustruje mogućnost beskonačne petlje u algoritmu; da bi ona bila izbegnuta, vrši se označavanje svakog poligona koji se pomera na kraj liste. Tako, kad god nijedan od prvih pet testova ne uspe i tekući poligon $Q$ je već obeležen, ne primenjujemo testove 3′ i 4′ već delimo $P$ ili $Q$ i u listu ih zamenjujemo njihovim delovima.

## 11.6 Rejkasting (raycasting) algoritam

- Appel (1968), Goldstein and Nagel (1968, 1971)
- Originalno razvijen za simuliranje putanja balističkih projektila, sada ima najznačajnije primene u grafici. Koristi se za određivanje vidljivosti, bulovske operacije nad telima, određivanje refleksije i refrakcije (prelamanja) itd.
- Umesto naziva *ray tracing* ponekad se koristi i naziv *ray casting*, mada se obično on koristi samo za varijantu za određivanje vidljivosti (a ray tracing za puni rekurzivni algoritam).
- Obično se koristi za off-line renderovanje
- Tipičan image-precision algoritam
- Puna, rekurzivna varijanta algoritma obrađuje i vidljivost i parametre kao senke, refleksije i refrakcije (prelamanja).
- Potrebno je zadati centar projekcije (*oko posmatrača*) i prozor proizvoljne ravni pogleda
- Može da se smatra da je prozor podeljen pravilnom mrežom (čiji elementi odgovaraju pikselima potrebne rezolucije)
- Za svaki elemenat mreže (tj. za svaki piksel) po jedan zrak se prati od centra projekcije, kroz sâm piksel do najbližeg objekta u sceni.
- Određuje vidljivost površina praćenjem imaginarnog zraka svetlosti od posmatračevog oka do objekta u sceni.
- Boja piksela je boja najbližeg objekta.




```
izaberi centar projekcije i prozor na projekcionoj ravni;
za svaku scan liniju na slici uradi sledece:
  za svaki piksel na scan 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
          zapamti presek i ime objekta;
      oboji piksel bojom najblizeg preseka
    end
```

- U osnovnoj verziji algoritma za svaki piksel $(x_1, y_1, z_1)$ potrebno je odrediti presečne tačke sa svim objektima tj. potrebno je odrediti vrednost parametra $t$ tako da tačka $x = x_0 + t(x_1 - x_0)$, $y = y_0 + t(y_1 - y_0)$, $z = z_0 + t(z_1 - z_0)$, pripada objektu; sva ta izračunavanja mora da budu brza.
- Za efikasnu primenu ray tracing algoritma koriste se mnoge optimizacije:
  - optimizacija izračunavanja preseka
  - izbegavanje (nepotrebnih) izračunavanja preseka (na bazi koherencije, neprekidnosti)
  - hijerarhije (objekat koji pripada nekom telu $V$ ne može da se seče sa zrakom svetla ako taj zrak ne seče telo $V$)
  - automatsko generisanje hijerarhija
  - deljenje prostora (najčešće pravilnom mrežom)
  - ...
- Postoje i varijante algoritma koje se bave problemom antialiasing (četiri zraka za svaki piksel, detektovanje i prikazivanje veoma malih objekata, itd)

Glava 12

# Nehromatska i hromatska svetlost

- Svetlost je značajna i interesantna tema za fiziku, psihologiju, kognitivne nauke, umetnost, dizajn, računarsku grafiku.
- Boja objekta ne zavisi samo od samog objekta već od izvora svetla koji ga obasjava, od boje okoline i od ljudskog vizuelnog sistema. Neki objekti (tj. materijali) reflektuju svetlost, dok neki propuštaju svetlost. Kod nekih objekata se javlja apsorbovanje, kod nekih razbacivanje svetlosti a kod nekih refrakcija (menja se pravac).
- Značaj boja u računarskoj grafici: estetski razlozi, realizam, naglašavanje, poboljšana funkcionalnost i interfejs.

## 12.1 Nehromatska svetlost

- Nehromatska (ili monohromatska) svetlost je npr. ono što vidimo na crno-belom televizoru/monitoru. Posmatrač nema osećaj za iskustvo koje se vezuje za boje (npr. za crvenu, plavu, zelenu).
- Kvantitet svetlosti je jedino svojstvo nehromatske boje.
- Kvantitet svetlosti može biti razmatran kao fizički pojam energije (kada se govori o *intenzitetu* ili *luminaciji*) ili kao psihološki fenomen (kada se govori o *sjajnosti* (eng. brightness)). Ovi koncepti su bliski i povezani, ali nisu identični.
- Crno-beli monitor može da proizvede mnogo različitih intenziteta svetlosti na poziciji jednog piksela. Mnogi uređaji (npr. štampači) mogu da proizvedu samo dva intenziteta boje (crna i bela). Određene tehnike omogućavaju ovakvim uređajima da simuliraju dodatne nivoe intenziteta.

## 12.2 Izbor intenziteta

- Pretpostavimo da hoćemo da reprezentujemo 256 intenziteta na skali od 0 do 1.

- Distribucija treba da bude ravnomerna u nekom smislu, ali treba voditi računa o sledećem: ljudsko oko je osetljivo na odnose intenziteta svetlosti, pre nego na apsolutne intenzitete svetlosti. Ljudskom oku se čini da se intenziteti 0.10 i 0.11 razliku isto kao i intenziteti 0.50 i 0.55. Dakle, intenzitet treba da bude opisan logaritamskom (a ne linearnom skalom) da bi se dobili jednaki koraci u sjajnosti.
- Da bi se odredilo 256 intenziteta počev od $I_0$ a do 1.0 potrebno je da postoji vrednost $r$ koja odražava odnos dva susedna intenziteta:

$$I_0 = I_0,\ I_1 = rI_0,\ I_2 = rI_1 = r^2 I_0,\ \ldots,\ I_{255} = r^{255} I_0 = 1$$

$$r = (1/I_0)^{1/255},\ I_j = r^j I_0 = I_0^{(255-j)/255}$$

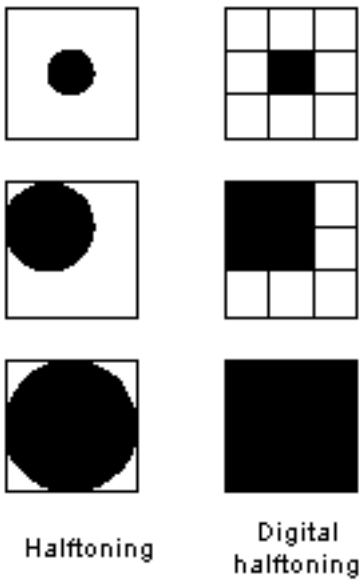
- Generalno (za $n$ intenziteta), važi:

$$r = (1/I_0)^{1/n},\ I_j = r^j I_0 = I_0^{(n-j)/n}$$

- Minimalni intenzitet $I_0$ je za CRT obično negde između $1/200$ i $1/40$ maksimalnog intenziteta.
- Prikazivanje opisanih intenziteta boja na CRT (ili na filmu) nije trivijalna stvar, zavise od konkretnog CRT uređaja i zahtevaju. Često se koriste unapred izračunate vrednosti (za određivanje napona tj. vrednosti za jedan piksel). Korišćenje ovakvih unapred izračunatih tabela se zove *gama korekcija* (eng. gamma correction)
- Ljudsko oko može da razlikuje susedne površine za koje je odnos intenziteta veći od 1.01.
- Na osnovu te vrednosti može se odrediti minimum intenziteta koji mogu da daju prihvatljivu ahromatsku sliku. Za većinu uređaja, ta (idealizovana) vrednost je 64.

## 12.3 Polutoniranje (halftoning)

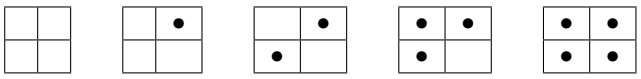
- Polutoniranje je tehnika koja na bazi minimalnog broja nivoa intenziteta (npr. crna i bela boja) daje privid više različitih nivoa osvetljenosti.
- Tehnika je veoma stara i nekada se koristila u bojenju tekstila i platna; moderno polutoniranje u štamparstvo je uveo Stephen Hargon 1880. godine.
- Tehnika se uspešno koristi npr. u štampanju novina sa niskom rezolucijom.
- Suštinski, tehnika se u različitim kontekstima (i na različitim medijima) koristi tako što intenzitet boje (a time osvetljenost i veličina obojene površine) zavisi od dužine izlaganju svetlu, mastilu i sl.
- Digitalno polutoniranje je slično polutoniranju u kojem je slika dekomponovana na mrežu polutoniranih ćelija. Nijanse boja na slici se simuliraju popunjavanjem odgovarajućeg broja ćelija. Naredna slika pokazuje kako se polutoniranje realizuje digitalno:



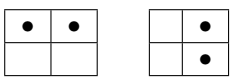

- Najčešći metodi polutoniranja su uzorkovanje, dithering i prostorni dithering (ili difuzija greške — error diffusion).

### 12.3.1 Uzorkovanje (patterning)

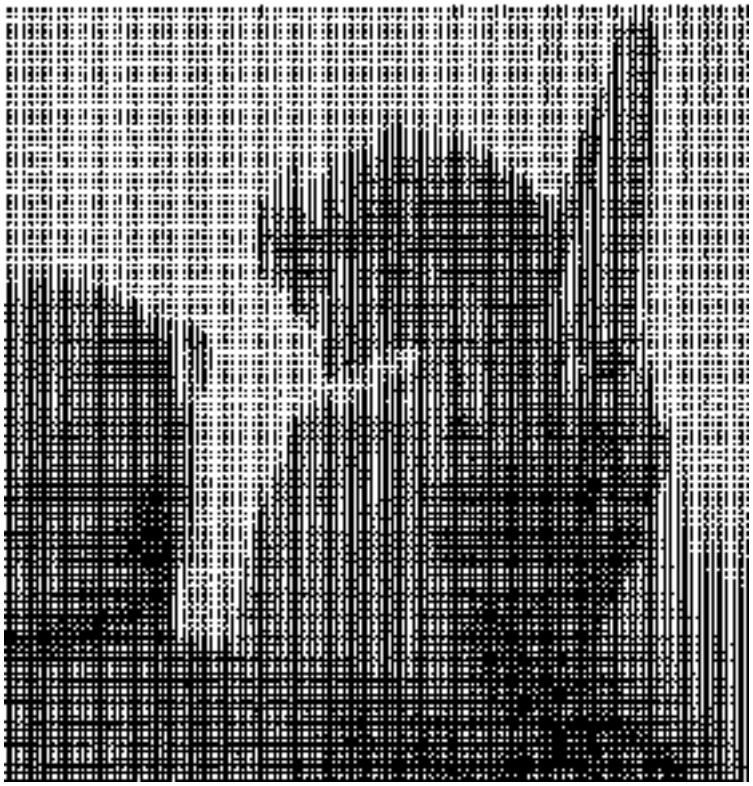
- Polutoniranje generalno može da koristi ćelije različite veličine. Za razliku od toga, uzorkovanje koristi ćelije jednake veličine.
- Pogodno korišćenje pojedinog uzorka samo od crnih i belih ćelija stvara privid nijanse sive.
- Uzorkovanje proizvodi sliku čije su dimenzije $n$ puta veće od polaznih dimenzija, gde je $n$ dimenzija ćelije.
- Uzorkovanje popravlja vizuelnu rezoluciju, ali loše utiče na prostornu (spatial) rezoluciju (gube se fini detalji) i zbog toga veličina uzorka/ćelije ne može da se uzima da bude prevelika.
- Za crnu i belu boju postoji 5 osnovnih $2\times2$ uzoraka (za 5 nijansi sive):



- ne smeju se koristiti sledeći uzorci (zbog vizuelnog utiska linija):



- za ćelije veličine $n$ treba da ima $n+1$ uzoraka
- Ilustracija korišćenja uzorkovanja za ćelije veličine $4\times 4$:



### 12.3.2 Dithering

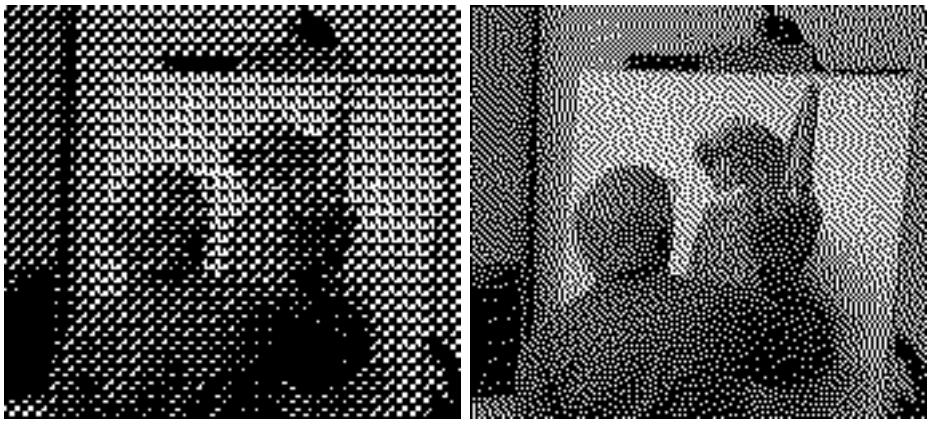
- *Dithering* (oklevanje, neodlučnost) je jedna od tehnika za digitalno polutoniranje.
- Dithering daje sliku istih dimenzija kao i polazna.
- Sistem obično automatski primenjuje ovu tehniku kada su karakteristike uređaja za prikaz podešene na 256 boja ili manje. Ova tehnika se često koristi da ublaži smenjivanje boja na slikama u GIF formatu.

- Dithering obično uvećava veličinu slike (u bajtovima), ali je poboljšan izgled vredan te cene.
- Dithering „razbacuje“ piksele različite boje po slici kako bi izgledalo da postoje međuboje na slici sa ograničenim brojem boja.
- Dithering poredi dither matricu sa delovima slike redom i za svaki piksel se proverava da li ima veću vrednost na slici ili u dither matrici (tzv. treshold vrednost). Kada ima veću vrednost na slici, onda se odgovarajući piksel u izlaznoj slici popunjava.
- Dithering može da koristi jasan uzorak ili slučajan šum.
- U prostornom ditheringu se greška koja nastaje za jedan piksel distribuira okolnim pikselima pre nego što se proverava treshold vrednost; postoji uzorak koji u tome daje optimalnu vrednost u smislu minimizovanje stvaranja efekta teksture).

Bez dithering-a (gif slika, 256 boja, 6Kb)

Dithering sa uzorkom (gif slika, 4 boje, 1.31Kb)

Dithering sa šumom (difuzija greške) (gif slika, 4 boje, 2.36Kb)



Dithering sa uzorkom i sa šumom

## 12.4 Hromatska (obojena) svetlost

- Vizualne senzacije uzrokovane obojenom svetlošću su mnogo bogatije nego one uzrokovane nehromatskom.
- Postoji više modela za opisivanje obojene svetlosti.
- Skoro svi modeli za opisivanje svetlosti zasnovani su na (nekim) trima veličinama.

### 12.4.1 CIE model

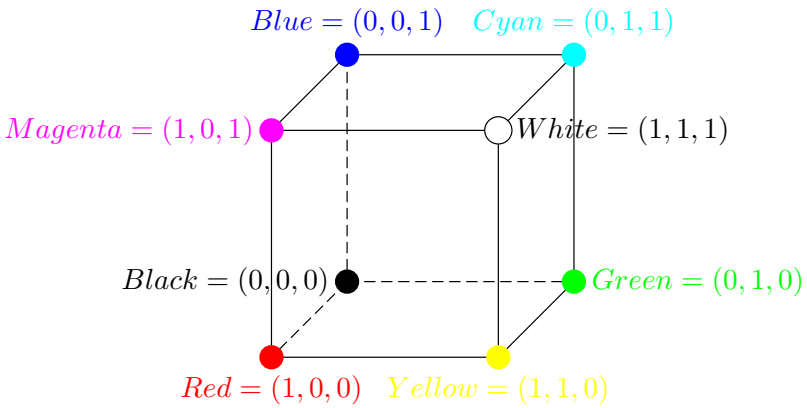
- 1931: Commission Internationale de l'Éclairage (CIE)
- Model zasnovan na tri osnovne vrednosti **X**, **Y** i **Z**. Ovim vrednostima odgovaraju funkcije $\bar{x}_\lambda$, $\bar{y}_\lambda$ i $\bar{z}_\lambda$.
- Kombinovanjem parametara **X**, **Y** i **Z** može se dobiti svaka vidljiva svetlost.

### 12.4.2 Modeli za rastersku grafiku

- Kolor model za rastersku grafiku je specifikovan 3D koordinatnim sistemom i podoblašću oblasti svih vidljivih boja (na primer, RGB kolor model je određen kockom).
- svrha kolor modela je da omogući jednostavno adresiranje boja iz nekog skupa (na primer, spektra CRT uređaja).
- Najčešće korišćeni modeli su
  - RGB i HSV (za CRT monitore)
  - YIQ (za TV kolor sistem)
  - CMY (za štampačke uređaje)

### 12.4.3 RGB kolor model

- Jedinična kocka sa temenima:
  - crvena (1,0,0)
  - zelena (0,1,0),
  - plava (0,0,1)




- RGB vrednosti se moraju interpretirati u zavisnosti od konkretnog uređaja (CRT monitora); za prelazak sa RGB komponenti na nove RGB komponente može se odrediti matrica (3x3)

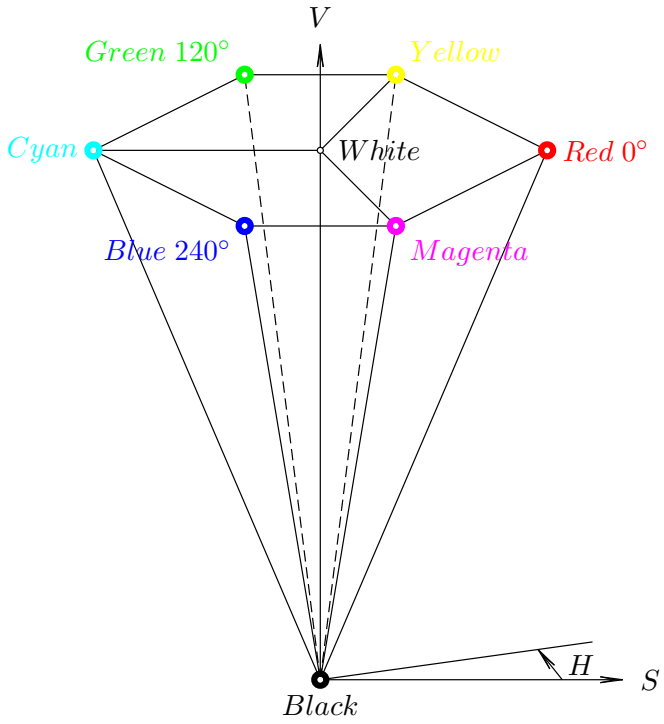
### 12.4.4 CMY kolor model

- Jedinična kocka sa temenima za cyan, magenta, yellow.
- U koordinatnom početku je bela umesto crne (kao u RGB modelu)
- Boje se specifikuju po tome što se „oduzima“ od bele, a ne po tome šta joj se dodaje; motivacija je u primeni ovog modela u štampačkim uređajima (na belom papiru nije moguće dobiti belu boju u RGB modelu kombinovanjem najviših intenziteta za crvenu, zelenu i plavu).
- Jednačina prelaska sa RGB komponenti na CMY komponente:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

### 12.4.5 HSV (HSB) kolor model

- Komponente:
  - hue (karakteriše boju) ($0° - 360°$, ili od 0 do 360)
  - saturation (zasićenost; karakteriše koliko se boja razlikuje od sive istog intenziteta) (0%-100%, ili od 0 do 240)
  - value (brightness; sjajnost) (0%-100%, ili od 0 do 240)




- Postoje jednostavni algoritmi (formule) za preračunavanje HSB u RGB komponente i obratno.

## 12.5 Izbor i korišćenje boja

- Interaktivno biranje boja.
- Interpoliranje (voditi računa o tome da linearnom prelazu između dve tačke u jednom modelu ne odgovara nužno linearni prelazak između dve tačke u drugom modelu).
- Redukcija boja (svođenje ukupnog broja boja na unapred zadat broj (pogodno odabranih) boja).
- Reprodukovanje boja.
- Boje u računarskoj grafici i računarstvu (estetski, funkcionalni, psihološki aspekti).

Glava 13

# Osvetljenje i senčenje (illumination and shading)

- Potreba za modeliranjem refleksija, propuštanja i prelamanja svetla, senki, određivanja boja tačaka objekata, tipova izvora svetla itd. Problem osvetljenja i senčenja je znatno komplikovaniji od problema vidljivosti. Uglavnom ćemo da razmatramo monohromatsku svetlost (tj. biće relevantan samo intenzitet svetla)

- Postoji više *modela osvetljenja* i *modela senčenja*. *model senčenja* je širi okvir i on koristi *model osvetljenja* (neki modeli senčenja pozivaju/koriste model osvetljenja za svaki piksel slike, dok drugi pozivaju model osvetljenja samo za neke piksele, dok se za preostale koristi interpolacija).

- Neka od rešenja za problem osvetljenja zasnovana su na iskustvu i eksperimentima i nisu utemeljena u fizici, ali daju dobre rezultate.

## 13.1 Modeli osvetljenja

### 13.1.1 Samoosvetljenost

- Najjednostavniji model osvetljenja je sledeći: za svaki objekat, svakoj tački je pridružen isti intenzitet svetlosti. U ovom nerealističnom modelu, svako telo kao da je samoosvetljeno.

- *Jednačina osvetljenja (*illumination*)* za ovaj model je

$$I = k_i$$

gde je $I$ rezultujući intenzitet svetla, a $k_i$ je intenzitet svetla pridružen objektu.

### 13.1.2 Ambijentalno svetlo

- *Ambijentalno svetlo* je difuzno, bez usmerenog izvora, koje je proizvod višestrukog odbijanja svetlosti od svih površina prisutnih u okruženju.

- Ako pretpostavimo da se svetlost rasprostire jednako u svim smerovima i po svim objektima, jednačina osvetljenja je za svaki objekat:

$$I = I_a k_a$$

gde je $I_a$ (konstantni) intenzitet ambijentalnog svetla, a $k_a$ je *koeficijent ambijentalne refleksije* objekta i on može da ima vrednosti između $0$ i $1$

### 13.1.3 Tačkasti izvor svetla i difuzna refleksija

- *Tačkasti izvor svetla* ravnomerno širi zrake u svim smerovima iz jedne tačke.
- *Difuzna ili lambertovska refleksija* refleksija (odbijanje svetlosti) od matiranih, hrapavih površina. Ovakve površine izgledaju jednako osvetljene iz svih uglova posmatranja i za jednu površinu osvetljenost zavisi samo od ugla $\theta$ između pravca svetla $\overline{L}$ i pravca normale $\overline{N}$.
- Jednačina osvetljenosti u ovom modelu je:

$$I = I_p k_p \cos\theta$$

gde je $I_p$ jačina izvora svetlosti, $k_p$ koeficijent difuzne refleksije (između $0$ i $1$) za materijal od kojeg je načinjen objekat i $\theta$ je ugao pravca svetla $\overline{L}$ i pravca normale $\overline{N}$.

- ako su vektori $\overline{L}$ i $\overline{N}$ normalizovani, onda je

$$I = I_p k_p (\overline{L} \cdot \overline{N})$$

- Ukoliko je izvor svetla dovoljno udaljen od svih objekata, možemo da smatramo da je vektor $\overline{L}$ konstantan za sve objekte i takav izvor svetla zovemo *direkcioni izvor svetla*
- Ukoliko se osvetljenost računa po fromuli

$$I = I_p k_p (\overline{L} \cdot \overline{N})$$

u dobijenom prikazu objekti izgledaju kao da su osvetljeni samo sa jedne strane i da se nalaze u mračnoj prostoriji (tj. zanemaruje se ambijentalna komponenta svetla). Da bi se taj efekat ublažio, često se koristi sledeća jednačina (koja uključuje i ambijentalna osvetljenost):

$$I = I_a k_a + I_p k_p (\overline{L} \cdot \overline{N})$$

- Ukoliko se projekcije dve paralelne površine od istog materijala preklapaju na slici, neće moći da se odredi gde prestaje jedna, a počinje druga (jer su projekcije iste boje). Da bi se razrešio ovaj problem uzima se u razmatranje i tzv. faktor *slabljenja izvora svetla* (*light-source attenuation factor*) $f_{att}$ uz koji jednačina osvetljenosti postaje:

$$I = I_a k_a + f_{att} I_p k_p (\overline{L} \cdot \overline{N})$$

- Najjednostavnija forma za $f_{att}$ je

$$f_{att} = \frac{1}{d_L^2}$$

gde je $d_L$ rastojanje od objekta do izvora svetla. Međutim, kada je izvor svetla daleko, ovaj model ne daje željeni efekat.

- Finiji model je opisan jednačinom:

$$f_{att} = \frac{1}{c_1 + c_2 d_L + c_3 d_L^2}$$

gde su $c_1$, $c_2$ i $c_3$ korisnički definisane konstante pridružene izvoru svetla. Često se ovaj model koristi kada je izvor svetla u tački pogleda.

- U razmatranje treba uvesti i rastojanje posmatrača do objekta.

### 13.1.4 Atmosfersko slabljenje

- Analogno kao što osvetljenost treba da slabi sa udaljenošću izvora svetla, tako treba da slabi i sa udaljenošću posmatrača.
- Prednjoj i zadnjoj ravni projektovanja pridružuju se faktori skaliranja $s_1$ i $s_2$, a faktor $s_o$ se određuje u zavisnosti od rastojanja između objekta i posmatrača (tj. u zavisnosti od $z$ koordinate). Ukoliko je $z_o$ (vrednost $z$ koordinate objekta) jednako $z_1$, onda je $s_o = s_1$ i, slično, ukoliko je $z_o = z_2$, onda je $s_o = s_2$. U opštem slučaju je

$$s_o = s_2 + \frac{(z_o - z_2)(s_1 - s_2)}{z_1 - z_2}$$

- Cilj je za svako $s_o$ odrediti prirodno $I'$ između (izračunate) osvetljenosti $I$ i osvetljenosti za udaljene objekte $I_{dc}$ (*depth-cue value*):

$$I' = s_o I + (1 - s_o) I_{dc}$$

### 13.1.5 Spekularna refleksija

- Spekularna refleksija može se videti na svakom glatkom, sjajnom objektu (npr. deo jabuke je najsvetliji, dok je ostatak obasjan difuznim svetlom; u tom delu jabuka ima belu, a ne crvenu boju)
- Spekularna refleksija javlja se samo u pravcu $\overline{R}$ koji je simetričan pravcu svetlosti $\overline{L}$ u odnosu na normalu površi $\overline{N}$.
- U Phong-ovom modelu, jednačina ukupne osvetljenosti je:

$$I = I_a k_a + f_{att} I_p (k_p \cos\theta + W(\theta) \cos^n \alpha)$$

gde je $\theta$ ugao između $\overline{L}$ i $\overline{N}$, $\alpha$ ugao između $\overline{R}$ i $\overline{V}$ (pravac gledanja), $n$ je *eksponent spekularne refleksije* za materijal (obično od 1 do nekoliko stotina), a za $W(\theta)$ se obično uzima konstantna vrednost $k_s$ — koeficijent spekularne refleksije (između 0 i 1).

- Ako su vektori $\overline{L}$, $\overline{N}$, $\overline{R}$ i $\overline{V}$ normalizovani, onda važi:

$$I = I_a k_a + f_{att} I_p (k_p (\overline{L} \cdot \overline{N}) + k_s (\overline{R} \cdot \overline{V})^n)$$

### 13.1.6 Prošireni izvori svetla (extended light sources)

- Za raliku od tačkastih izvora svetla, *prošireni* ili *distribuirani* izvori svetla imaju površinu i, kao posledicu, daju mekše senke.

## 13.2 Hromatska svetlost

- Prethodni modeli i jednačine razmatrali su samo monohromatsku svetlost. Hromatska svetlost se obično obrađuje tako što se posebno obrađuju jednačine za njene tri komponente.
- Na primer, $(O_{dR}, O_{dG}, O_{dB})$ definiše za objekat difuzno crvenu, zelenu i plavu komponentu u RGB kolor sistemu.
- Na primer, u kolor modelu jednačina

  $$I = I_a k_a + f_{att} I_p k_d (\overline{L} \cdot \overline{N})$$

  daje

  $$I_R = I_{aR} k_a O_{dR} + f_{att} I_{pR} k_d O_{dR} (\overline{L} \cdot \overline{N})$$

- Idealno, kolor senčenje bi trebalo vršiti kombinovanjem (neprekidnih) rezultata za sve boje iz spektra. No, i opisani simplifikovani model obično daje prihvatljive rezultate.
- Na primer, u zavisnosti od talasne dužine Phongova jednačina

  $$I = I_a k_a + f_{att} I_p (k_p (\overline{L} \cdot \overline{N}) + k_s (\overline{R} \cdot \overline{V})^n)$$

  dobija finiji oblik:

  $$I_\lambda = I_{a\lambda} k_a O_{d\lambda} + f_{att} I_{p\lambda} (k_d O_{d\lambda} (\overline{L} \cdot \overline{N}) + k_s O_{s\lambda} (\overline{R} \cdot \overline{V})^n)$$

  gde je $O_{s\lambda}$ spekularna boja objekta.

## 13.3 Senčenje za poligone

- Senčenje poligona može da koristi razne specifičnosti modela.
- Najjednostavnija varijanta je ona sa sledećim pretpostavkama:
  - Izvor svetla je beskonačno daleka tačka, pa je vrednost $\overline{L} \cdot \overline{N}$ konstantna za sve tačke jednog poligona
  - Tačka posmatranja je beskonačno daleka, pa je vrednost $\overline{V} \cdot \overline{N}$ konstantna za sve tačke jednog poligona
  - Poligoni odgovaraju stvarnom objektu i nisu njegova aproksimacija
- Ako su vrednosti $\overline{L}$ i $\overline{N}$ konstantne, konstantna je i vrednost $\overline{R}$ (jer je vektor $\overline{R}$ simetričan vektoru $\overline{L}$ u odnosu na $\overline{N}$), pa je konstantna i vrednost $\overline{R} \cdot \overline{V}$.
- Model je pogodan i za optimizacije zasnovane na interpolaciji i sl.

## 13.4 Senke

- Algoritmi za vidljivost određuju koji delovi površi se mogu videti iz tačke posmatranja, dok algoritmi za senke određuju koji delovi površi se mogu videti iz izvora svetla. Oni delovi površi koji se ne vide iz izvora svetla su u senci.

- Vidljivost iz tačkastog izvora svetla je, kao i vidljivost iz tačke posmatranja — sve ili ništa.

- Ako je dato više izvora svetla ($m$), Phongova jednačina osvetljenosti postaje:

$$I_\lambda = I_{a\lambda} k_a O_{d\lambda} + \sum_{1 \le i \le m} S_i f_{att} I_{p\lambda} (k_d O_{d\lambda} (\overline{L} \cdot \overline{N}) + k_s O_{s\lambda} (\overline{R} \cdot \overline{V})^n)$$

  gde je $S_i$ jednako $0$ ako je svetlost $i$ blokirana u ovoj tački, a jednako $1$ inače.

## 13.5 Transparentnost

- Neki objekti/materijali propuštaju deo svetlosti; obično se ta svetlost *prelama* (*refract*), ali se taj efekat često zanemaruje.

- Modeli u kojima se transparentnost obrađuje ne vodeći računa o prelamanju:

  **interpolirana transparentnost:** Ukoliko je poligon $P_1$ transparentan i nalazi se ispred poligona $P_2$, intenzitet svetlosti je jednak:

$$I_\lambda = (1 - k_{t_1}) I'_\lambda + k_{t_1} I''_\lambda$$

  gde je $I'_\lambda$ svetlost za poligon $P_1$, $I''_\lambda$ za poligon $P_2$, a $k_{t_1}$ *koeficijent transmisije* za poligon $P_1$ i on predstavlja meru transparentnosti. 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.

  **filtrirana transparentnost:** poligon se tretira kao filter koji selektivno propušta različite talasne dužine:

$$I_\lambda = I'_\lambda + k_{t_1} O_{t\lambda} I'_\lambda$$

  gde je $O_{t\lambda}$ boja transparencije (i može biti posebno zadavana za svaku vrednost $\lambda$)

- Prelamanje (engl. refraction) svetlosti opisuje jednakost

$$\frac{\sin \theta_1}{\sin \theta_2} = \frac{\nu_1}{\nu_2}$$

  gde su $\nu_1$ i $\nu_2$ indeksi refrakcije materijala kroz koje svetlost prolazi.

## 13.6 Međuobjektne refleksije i globalna iluminacija (osvetljenje)

- Međuobjektne refleksije se javljaju kada se na površini objekta reflektuju druge površine u okruženju. Ovaj efekat može da se kreće od spekularne refleksije (koja se menja sa položajem posmatrača), pa do difuzne refleksije (na koje ne utiče pozicija posmatrača).

- Modeli osvetljenja računaju boju u tački u terminima svetlosti direktno emitovane od strane izvora svetla i terminima svetlosti koja dolazi do tačke nakon reflektovanja i transmisije kroz razne površi. Ova indirektno reflektovana i transmitovana svetlost se obično zove *globalno osvetljenje* ili *globalna iluminacija*. (*Lokalna iluminacija* je svetlost koja dolazi direktno iz izvora svetla do tačke koja se senči.

### 13.6.1 Rekurzivni ray-tracing algoritam

- Osnovni ray-tracing algoritam za vidljivost određuje tačku preseka koja je najbliža posmatraču (i njenu boju). Da bismo odredili senke, upućujemo dodatni zrak svetlosti iz te tačke preseka do svakog izvora svetla. Ako neki od ovih *zraka senke* preseca neki objekat duž svog puta, onda je polazna tačka u senci i taj izvor svetla ne doprinosi njenoj boji.

```
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 if 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.
```

Glava 14

# Vizuelni realizam

U prethodnom tekstu korišćeni su prilično jednostavni modeli, zasnovani na linijama, poliedrima i jednostavnim krivama. Foto-realističan prikaz zahteva dodatne tehnike (i ,,trikove“) koji mogu da zavaraju posmatrača i stvore mu privid stvarne scene.

Za visok kvalitet slike može se koristiti veliki broj poligona, i to daje odlične rezultate za neprirodne, ali ne uvek i za prirodne objekte.

Algoritmi za određivanje vidljivih površi (na primer, zasnovani na z-bafer algoritmu) i dobri algoritmi za senčenje mogu da daju veoma realistične slike. Postoje, međutim mnogi problemi sa renderovanjem vizuelno realističnih slika:

- teksture (npr. koža, zid, trava);
- suptilne promene boje (drvo, zid, nebo);
- senke;
- refleksije (metalni objekti, staklo);
- nepravilnosti (trava, oblaci);

Čuvanje mnogo detalja za ovakve objekte može da zahteva previše resursa, ali bez toga scena izgleda veštački.

Metodi kojima se popravlja vizuelni realizam uključuju:

- anti-aliasing (zbog njenog značaja ova tehnika se primenjuje u skoro svim bibliotekama i grafičkim karticama);
- preslikavanje tekstura (tekstura se preslikava na vidljivu površ, to može da bude komplikovano);
- preslikavanje ,,izbočina“ (često se koristi sa preslikavanjem tekstura; preslikavanje ,,izbočina“ koristi preslikavanje specifičnih delova na poligone tako da se njihov izgled menja u zavisnosti od svetla);
- boja (na primer, objekti u daljini izgledaju plavlje);

- svojstva materijala (količina i tip svetlosti koja se reflektuje, prelama i propušta kroz objekat može biti opisana na razne načine;
- korišćenje krivih i površi (one mogu izgledati znatno bolje nego mreže poligona);
- bolje modelovanje izvora svetla (na primer, neki izvori svetla ne daju oštre senke);
- napredno modelovanje kamere (tj. optičkog sistema) (i ljudsko oko i fotografija ne vide savršeno na svim daljinama; može se simulirati da su neki delovi van fokusa ili zamućenost pokretom koja je posledica duže ekspozicije);
- podešavanje daljine (utisak daljine se može postići smanjivajnem intenziteta slika kao funkcijom daljine).

Jedan od pristupa računarske grafike je stereovizija (3D prikaz). Umesto projketovanja jedne 2D verzije 3D objekta, proizvode se dve 2D verzije, po jedna za svako oko, uzimajući u obzir različite pozicije očiju,. Ovaj pristup je ključan u aplikacijama koje uključuju takozvanu virtuelnu stvarnost. Postoje različite softverske i hardverske tehnike za ovaj pristup.

Glava 15

# Manipulacija slikama

Deo računarske grafike je i problematika manipulacije (2D) slikama — njihovo pohranjivanje (tj. kompresija) i obrađivanje.

## 15.1 Kompresija slika

Slika veličine 1024x768 piksela u true-color sistemu (24 bita) zauzima 2.4Mb (u nekomprimovanom obliku, npr. .bmp). To danas često nije kritično zahvaljujući jeftinim hard diskovima velikih kapaciteta, ali iz mnogo razloga je potrebno da slike ne zauzimaju previše prostora. Postoje dva osnovna pristupa za kompresiju slika:

- *lossless* kompresija (kompresija bez gubitka informacija);
- *lossy* kompresija (kompresija sa gubitkom informacija);

### 15.1.1 Lossless kompresija

Lossless kompresija se koristi i za komprimovanje drugih podataka, ne samo slika. Ipak, mnogi lossless algoritmi su prilagođeni za komprimovanje slika i koriste specifičnosti ovog domena.

Kada se komprimuje slika, pogodno ju je konvertovati iz RGB prostora u HSV prostor jer na slikama R, G, i B komponente mogu veoma da variraju, dok se u HSV modelu često samo V komponenta ima veliku raznolikost.

Stepen kompresije koji se može dobiti zavisi od izabranog metoda i složenosti slike. Jedna mera složenosti slike je *entropija*. Entropija je mera neuređenosti ili slučajnosti. Slika koja se sastoji od slučajno izabranih piksela ima maksimalnu entropiju, dok slika koja se sastoji od samo jedne boje ima minimalnu entropiju.

Lossless kompresija može da daje stepen kompresije 3:1, što je često premalo, posebno ako nije bitno na slici zadržati sve informacije.

Neke od tehnika za lossless kompresiju su:

**Kodiranje po dužini.** Čuva se broj pojavljivanja jedne vrednosti u nizu (i počiva na koherentnosti linija slike). Na primer, niz:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 63 | 63 | 63 | 63 | 64 | 64 | 64 | 78 | 89 | 89 | 89 | 89 |

se može predstaviti kao 63,4,64,3,78,1,89,4. Kompresija daje najbolje rezultate (stepen kompresije i više od 4) na binarnim (crno-belim) slikama. Ukoliko su sve (ili mnoge) uzastopne vrednosti u nizu različite ova tehnika daje duži niz nego originalni.

**Hafmanovo kodiranje.** Hafmanovo kodiranje koristi histogram svetlosti za sliku koja se komprimuje. Svakom obrascu (od jednog ili više bajtova) u nizu (u originalnom zapisu slike) daje se kraći kôd ukoliko se on pojavljuje češće (i duži kôd ukoliko se on pojavljuje ređe). Hafmanovo kodiranje može se primeniti i nakon kodiranja po dužini radi još boljih rezultata.

**Prediktivno kodiranje.** I prediktivno kodiranje počiva na koherentnosti linije slike (razlike između susednih tačaka ne postoje ili su male) i pohranjuje vrednost prvog piksela i onda razliku u odnosu na sledeći piksel (za koju se očekuje da će moći da bude pohranjena manjim brojem bitova). Na primer, niz:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 63 | 63 | 63 | 63 | 64 | 64 | 64 | 78 | 89 | 89 | 89 | 89 |

se može zapisati u obliku: 63,0,0,0,1,0,0,14,11,0,0,0.

Ukoliko broj bitova predviđen za razlike nije dovoljan, koriste se specijalne overload obrasci.

Tehnika se može primenjivati i u više iteracija (jer razlike drugog reda mogu da budu još manje od razlika prvog reda).

**Blokovsko kodiranje.** Blokovsko kodiranje traži na slici blokovske obrasce, ne samo obrasce duž linija. Ovo je vremenski zahtevnije od prethodnih algoritama, ali daje nešto bolje stepene kompresije.

Lemle-Ziv-Welch kodiranje je varijanta blokovskog kodiranja koje se koriši u .gif formatu.

### 15.1.2 Lossy kompresija

Lossy algoritmi se primenjuje ako je važno imati što veći stepen kompresije i ako nije bitno na slici zadržati sve informacije. U lossy algoritmima često se može birati stepen kompresije (čime se, naravno, diktira kvalitet slike). I uz veliki stepen kompresije kvalitet slike je zadovoljavajuć. Cena ovih pogodnosti lossy algoritama je njihova vremenska zahtevnost.

Neke od tehnika za lossy kompresiju su:

**Kodiranje odsecanjem.** Ova tehnika uklanja bitove najmanje težine. Na primer, ako u osmobitnoj slici koristimo samo po 4 bita najveća težine, nova slika će zauzimati dva puta manje prostora. Za rekonstrukciju čitave slike koriste se tehnike interpolacije i dodavanja šûma (kako bi se izbegao efekat postera — velikih blokova iste boje). Kodiranje odsecanjem koristi se i za nizove slika, jednostavno izbacivanjem, na primer, svakog drugog frejma

**Lossy prediktivno kodiranje.** Lossy prediktivno kodiranje liči na lossless prediktivno kodiranje, s tim da se ne koriste se specijalne overload obrasci (ukoliko broj bitova predviđen za razlike nije dovoljan). Ukoliko broj

bitova predviđen za razlike nije dovoljan, ta razlika se propagira na naredne piksele. Ovo će dati umekšane ivice ali će da popravi stepen kompresije. Na primer, niz

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 63 | 63 | 63 | 63 | 64 | 64 | 64 | 78 | 89 | 89 | 89 | 89 |

se može reprezentovati kao: 63,0,0,0,1,0,0,8,8,8,1,0, a odavde rekonstruisan niz je:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 63 | 63 | 63 | 63 | 64 | 64 | 64 | 72 | 80 | 88 | 89 | 89 |

**Lossy blokovsko kodiranje.** Lossy blokovsko kodiranje, kao i lossless blokovsko kodiranje, obrađuje blokove slike (i koristi predefinisane obrasce), ali ne nužno egzaktno. Minimizuje se razlika između originalne i rekonstruisane slike. Često se, kao mera razlike koristi suma kvadrata razlika sjajnosti piksela (jer je ljudsko oko najosetljivije na sjajnost).

Fraktalna kompresija koristi i predefinisane blokove i slika se opisuje pomoću translacija, rotacija i skaliranja ovih osnovnih obrazaca.

**Transform kodiranje.** Transform kodiranje je slično blokovskom kodiranju, ali su ovde blokovi predefinisani (obično dimenzija 4x4 ili 8x8). Za obradu blokova koristi se 2D Furijeova transformacija.

Druga, sve popularnija metoda zasniva se na *talasićima* (*wavelet* transformacijama) koje koriste nešto drugačije osnovne funkcije za opisivanje slike. Ove funkcije se mogu podesiti konkretnoj slici. Često stepen kompresije sa odličnim kvalitetom može da bude 10:1 i više. Cena tako velike komresije je vremenska zahtevnost.

U ovoj grupi je `.jpg` kodiranje koje pored osnovne transformacije koristi i lossless prediktivno i Hafmanovo kodiranje. Kvalitet i veličuna slike mogu se zadati izborom paremetra.

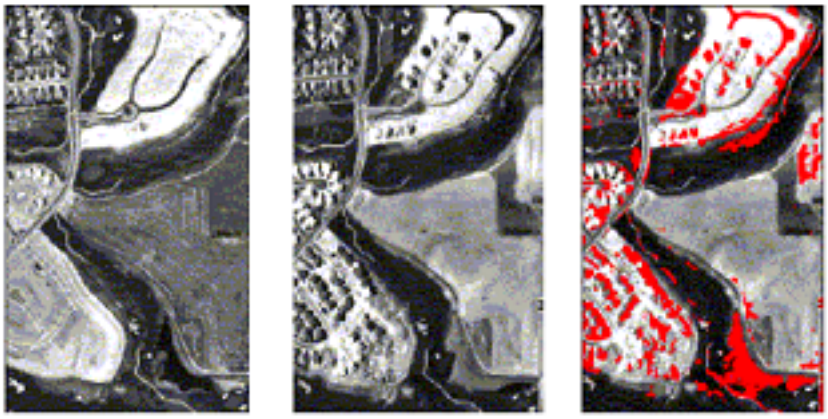
## 15.2 Obrada slika

Jedna od prvih znavčajnijih primena obrade slike bila je, šezdesetih godina dvadesetog vec, analiza slika Mesečeve površine kako bi se odredilo pogodno mesto za sletanje prvih letelica.

Neke od primena obrade slika su:

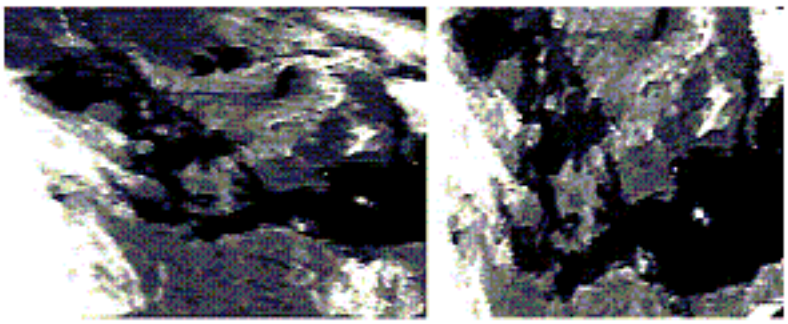
- analiza predmeta na daljinu;
- medicinske primene;
- obaveštajne primene;
- bezbednosno nadgledanje;
- obrada dokumenata;
- umetnost;
- navođenje kretanja (automobila i sl).

Sledeći primer ilustruje jednu primernu obrade slike — u detektovanju urbanističkih promena između dva snimanja:



Tehnike za obradu slika mogu se grubo podeliti na:

- tehnike za unapređivanje slika;
- tehnike za rekonstrukciju slika — tehnike za ispravljanje grešaka nastalih zbog svojstva kamere (photogrammetric correction) ili pogleda (geometric correction). Sledeći primer ilustruje geometrijsku korekciju satelitskog snimka Danske i Švedske.

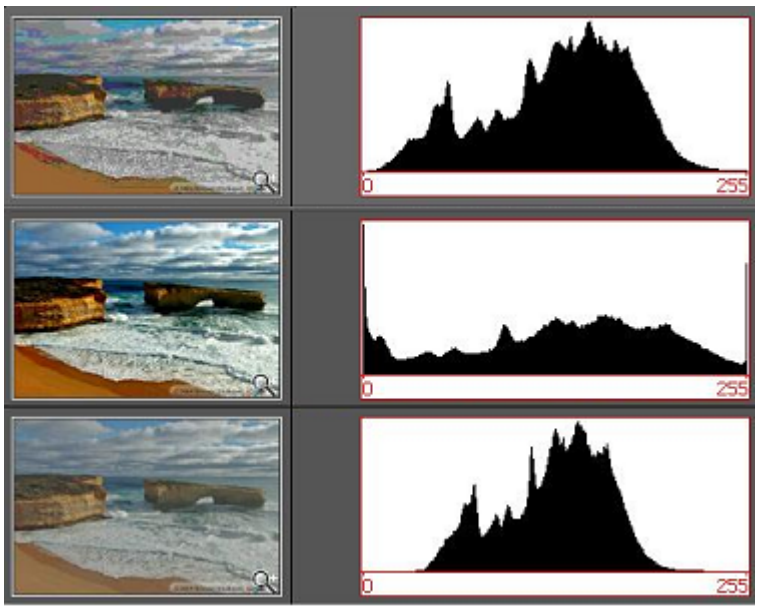


## 15.3 Unapređivanje slika

U okviru unapređivanja slika neke od centralnih tema su:

**Popravljanje kontrasta.** Cilj popravljanja kontrasta je povećavanje ili smanjivanje svetlosnog kontrasta između delova slike. Ova operacija se definiše nad histogramom slike, a ne nad njenim pojedinačnim pikselima. Histogram slike je histogram vrednosti intenziteta svetlosti. Za kolor slike postoje tri histograma, po jedan za svaki kanal (RGB ili HSV).

Kontrast se uvećava tako što se histrogram ,,razvlači" ka granicama, a umanjuje ako se skuplja ka sredini. Ovo je ilustrovano sledećim primerom.




**Filterisanje slike.** Operacije filterisanja obično deluju lokalno. One mogu da, na primer, omekšavaju ili pooštravaju sliku.

Filter za omekšavanje (*smooth* ili *blur*) može biti opisan na sledeći način:

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Ova transformacija primenjuje se na sve 3x3 delove slike i zadržava ukupno osvetljenost slike nepromenjenu (ali ona menja elemente sa visokim ili niskim intenzitetor).

Filter pooštravanja može biti opisan na sledeći način:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Postoje i mnogi drugi filteri — za druge vrste omekšavanja i pooštravanja, za isticanje ivica itd.

*Appendix* $A$

# Primeri testova i ispitnih zadataka

## A.1 Računarska grafika, I smer, prvi test, 26.11.2007.

1. Opisati ukratko sisteme izlazne tehnologije za računarsku grafiku. Nabrojati vrste štampača.

2. Šta je to adresivost, rezolucija, a šta rastojanje između tačaka?

3. Napisati midpoint algoritam za crtanje kruga (verzija koja koristi razlike drugog reda) i ukratko ga objasniti.

4. Opisati scan algoritam za popunjavnje poligona.

5. Opisati Cohen-Sutherland-ov algoritam za kliping linija.

6. Šta je to aliasing, a šta antialiasing? Koja su dva osnovna metoda za antialiasing (objasniti ih ukratko)?

7. Šta je to polutoniranje i koje su osnovne metode polutoniranja (objasniti ih ukratko)?

8. Opisati matricom transformaciju koja prevodi pravougaonik sa donjim levim uglom (-1,-1) i gornjim desnim uglom (1,1) u 2D koordinatnom sistemu $(u, v)$, u pravougaonik sa donjim levim uglom (2,1) i gornjim desnim uglom (4,2) u 2D koordinatnom sistemu $(x, y)$.

9. Navesti tipove 3D planarnih projekcija.

10. Šta čini VRC (view reference coordinate system), kako se on konstruiše i čemu služi?

## A.2 Računarska grafika, završni ispit, januar 2008

Parametarska jednačina eliptičkog paraboloida se može predstaviti sledećim formulama:

$$x = a\sqrt{u}\cos v$$

$$y = b\sqrt{u}\sin v$$

$$z = u$$

gde $u \in [0, \infty)$, $v \in [0, 2\pi]$, a $a$ i $b$ predstavljaju veliki i mali poluprečnik elipse. Napisati program koji, koristeći *display liste*, *vertex* polja i polja normala, iscrtava eliptički paraboloid dat gore navedenim jednačinama.

Uključiti jedan izvor svetlosti čije parametre treba postaviti na razumne vrednosti. Pozicija izvora svetlosti neka bude u tački u kojoj se nalazi *posmatrač*.

Napisati funkciju `void init(void)`, koja će alocirati dovoljno memorije za smeštanje podataka o čvorovima koji će biti korišćeni pri iscrtavanju i o normalama u tim tačkama. Na osnovu gore navedenih jednačina, popuniti nizove čvorova i normala. Omogućiti korišćenje *OpenGL*-ovih internih nizova i popuniti ih izračunatim podacima. Generisati listu za prikazivanje i u njoj *opisati* način generisanja paraboloida. Gledajući odozdo nagore, prvi *red* treba iscrtati kao `GL_TRIANGLE_FAN`, a sve ostale kao `GL_QUAD_STRIP`. Funkciju `init()` treba pozvati u `main`-u pre poziva `glutMainLoop()`, a u *display callback* funkciji treba pozvati listu za prikaz pomoću `glCallList()`.

Parametar $u$ ograničiti odozgo, tj. uvesti novu promenljivu $H$ koja će predstavljati *visinu* paraboloida (tj. $u \in [0, H]$). U kodu deklarisati promenljive $nu$ i $nv$ koje će predstavljati broj podela po odgovarajućem parametru.

Osim ovoga, omogućiti rotaciju oko odgovarajuće ose, u zavisnosti kako pomeramo miša.

Omogućiti da se pritiskom na taster $m$ menja mod za iscrtavanje poligona (`GL_LINE`, `GL_POINT`, `GL_FILL`).

Neka parametri $a$ i $b$ budu u početku postavljeni na vrednost 1. Omogućiti da se parametar $a$ povećava i smanjuje za vrednost 0.1 svaki put kada se pritisne respektivno taster $a$ i $A$. Istu stvar uraditi i za drugi parametar. Voditi računa da kada se pritisne npr. taster $a$ nije dovoljno samo uvećati vrednost tog parametra i pozvati `glutPostRedisplay()` već je izmedju još jednom potrebno pozvati funkciju `init()` da bi se ponovo generisala lista (sa novom vrednošću parametra $a$).

Voditi računa o dinamički alociranoj memoriji, tj. oslobadjati je čim ne bude više potrebna. Takodje koristiti `glDelete*()` funkcije za uništavanje ostalih korišćenih objekata.

**Uputstvo za rad**

Prijavljivanje na računar se vrši korišćenjem korisničkog imena `ispit1` i lozinke `ispit1`. Grafičko okruženje se pokreće komandom `startx`. Pre početka rada u direktorijumu `/home/ispit1` treba napraviti poddirektorijum koji treba da ima isto ime kao što je korisničko ime studenta na *Alas*-u (npr. mi08001). Ukoliko još neki direktorijum sa imenom iste strukture postoji odmah ga treba izbrisati. U novonapravljeni direktorijum treba stavljati sve fajlove koji se odnose na rešenje zadatka. U direktorijumu mora da postoji odgovrajući `Makefile` koji će omogućiti automatsko prevodjenje. `Makefile` treba da izgleda onako kako je objašnjeno na vežbama. U slučaju da se kod nalazi u fajlu `zadatak.c` a želimo da napravimo izvršni fajl sa imenom `zadatak` to ćemo postići komandom

```
gcc -L/usr/X11R6/lib -o zadatak zadatak.c -lglut
```

Ovako preveden program se izvršava komandom

```
./zadatak
```

Da bi zadatak bio bodovan, neophodno je da može da se prevede i pokrene i da se iscrtava bar neki deo onoga što je zahtevano u zadatku. Dokumentovanost rešenja, tj. postojanje odgovarajućih komentara u kodu, će biti takodje bodovana.