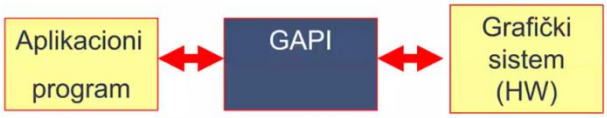
1. Šta je grafički API?

Grafički API je skup grafičkih f-ja organizovanih u jednu ili više biblioteka koje predstavljaju interfejs između aplikacionog programa i grafičkog sistema. To omogućava programeru da vidi samo grafički API, dok su detalji o softverskoj i hardverskoj implementaciji sadržani u grafičkoj biblioteci. Korišćenje grafičkog API-ja omogućava da se sistem posmatra kao „crna kutija“.

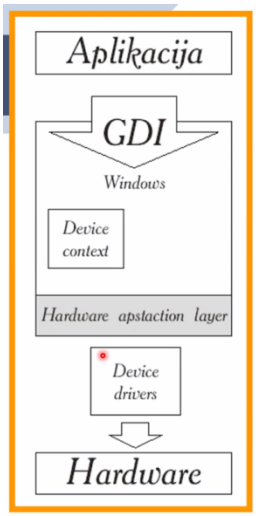


2. Funkcije GAPI-ja?

Da bi GAPI bio GAPI mora da ima sledeće grupe funkcija:

1. Funkcije za primitive – Grafičke primitive su objekti najnižeg nivoa, osnovni entiteti koje grafički sistem može da prikaže (tačka, linija, krug, elipsa, ...)
2. Funkcije za kontrolu atribut primitiva – Primitive definišu šta će se nacrtati, a njihovi atributi kako će se nacrtati (boja, debljina linije, tip linije)
3. Funkcije pogleda – Omogućavaju formiranje 2D prikaza na osnovu 3D objekata (koji objekti 3D sveta će biti uključeni u 2D prikaz i na koji način) - ekrani su 2D i sve što modelujemo u 3D mora da se iscrta u 2D ali tako da korisnik i dalje ima osećaj 3D prostora
4. Funkcije za geometrijske transformacije – Omogućavaju 2D i 3D geometrijske transformacije nad objektima (translacije, rotacije, skaliranje, smicanje, ...) – svodi se na množenje kordinata objekta nekim matricama
5. Funkcije za ulaz – Omogućavaju korisnicima da vrše unos podataka u aplikaciju putem delova grafičkog sistema (tastatura, miš, tabela, skener)
6. Kontrolne funkcije – Omogućavaju komunikaciju sa OS pod kojim se grafička aplikacija izvršava (inicijalizacija, kontrola prozora, otvaranje dialog box-a, postavljanje menija u prooru, ...)
7. Ispitivačke funkcije – Omogućavaju dobijanje informacija o karakteristikama pojedinih komponenata grafičkog sistema (broj boja koje monitor može da prikaže, rezolucija, ...). Na ovaj način programeri dobijaju mogućnost da pišu device independant aplikacije koje ne zavise od grafičkog sistema na kome se izvršavaju.

3. Princip rada GDI i šta je Device Context?



Sa jedne strane nalazi se aplikacija koju programer sam razvija, sa druge strane u formiranju slike učestvuje sam hardver (grafička kartica) odnosno device driver koji upravlja tom grafičkom karticom. Između njih nalazi se sloj grafičkog API-ja koji predstavlja sponu između aplikacije i drajvera, odnosno samog grafičkog hardvera. Osnovna stvar međusloja je GDI tj. Grafički API za 2D grafiku. Ako posmatramo detaljnije, pre same komunikacije GDI-ja sa device driver-ima nailazimona sloj hardverske apstrakcije koji sadrži funkcije operativnog sistema koje su te koje direktno interaguju sa drajverima uređaja.

Device context je apstrakcija koja olakšava rad sa grafičkim hardverom, to je apstraktni (virtuelni) uređaj koji se ne odnosi ni na jedan konkretan uređaj – ni na ekran, ni na štampač, ... Kada se programira, pristupa se ovom virtuelnom uređaju i na taj način se ne vodi računa direktno o hardveru. Svo iscrtavanje se vrši po Device Context-u, koji se kasnije, kroz funkcije GAPI-ja, može vezati za konkretan izlazni uređaj kako bi se ono što je iscrtano u DC prikazalo na tom konkretnom uređaju. O upravljanju samog isctavanja po izlaznom uređaju vode računa drajveri i hardware abstraction layer.

4. Objasniti strukturu OpenGL biblioteke

OpenGL je organizovan u nekoliko biblioteka:

- **GL (Graphics Library)** je jezgro OpenGL-a. Ona je osnovna biblioteka OpenGL-a i sadrži imena svih OpenGL funkcija.
- **GLU (Graphic Utility Library)** je biblioteka koja obezbeđuje potrebnu funkcionalnost OpenGL jezgra. Ona koristi samo GL funkcije i sadrži kod za kreiranje objekata koji se često koriste (npr. lopta, prizme, krive, površi, ...) kao i procedure koje se često koriste.
- **GLUT (GL Utility Toolkit**) je biblioteka koja obezbešuje minimum funkcionalnosti koja se očekuje od modernih sistema zasnovanih na prozorima (otvaranje prozora, ulaz – miš, tastatura, meniji, ...). Obezbeđuje kontrolne funkcionalnosti GAPI-ja.
- **GLX (WGL, AGL)** su biblioteke koje služe za spregu sa operativnim sistemima. GLX je biblioteka koja se koristi za Unix/Linux based operativne sisteme, WGL za Windows i AGL za Apple/MacOS. Korišćenjem funkcija ovih biblioteka gubi se potreba da OpenGL aplikacija poziva direktno funkcije operativnog sistema.

5. Čemu služe ulazni, a čemu izlazni uređaji

Ulazno/izazni uređaji su značajni za računarsku grafiku jer obezbeđuju interakciju sa spoljnim svetom. Ulazni uređaji služe za upravljanje računarom, prihvataju podatke od korisnika i konvertuju ih u formu koju ralunar može da razume. Izlazni uređaji služe za praćenje rada računara kao i za prikazivanje rezultata obrade u formi razumljivoj za korisnika.

6. Logički tipovi U/I uređaja

1. Lokatori – pokazuju poziciju i orijentaciju – grafička tabla, miš, džojstik, ekrani osetljivi na dodir
2. Valuator – za unos jednog realnog broja – potenciometar
3. Tastatura – za unos znakovnog niza
4. Pokazivač – za izbor elemenata slike – svetlosno pero (stilus)
5. Dugme – za izbor neke akcije iz skupa mogućih alternativa
6. Uređaji za unos slike, videa i 3D modela – 2D skener, digitalni fotoaparati, digitalne i web kamere i 3D skeneri
7. Uređaji za unos zvuka – mikrofoni, digitalni diktafoni i digitalni snimači zvuka
8. VR ulazni uređaji – VR rukavice i različiti uređaji za detekciju ljudskih pokreta

7. Razlika između vektorskih i rasterskih podataka

Kod vektorskih podataka svi objekti koji se prikazuju se zadaju koordinatama svojih karakterističnih tačaka koje omeđuju taj objekat. Za razliku od njih, kod rasterskih podataka se svi objekti koji se prikazuju zadaju kao raster, odnosno matrica tj. Skup piksela. Jedini objekat kod rasterskih podataka je piksel, odnosno tačka. Ti pikseli organizovani su u kolone i vrste i svaki ima svoju boju. Posmatranjem rastera u celosti pojedinačni pikseli formiraju sliku.

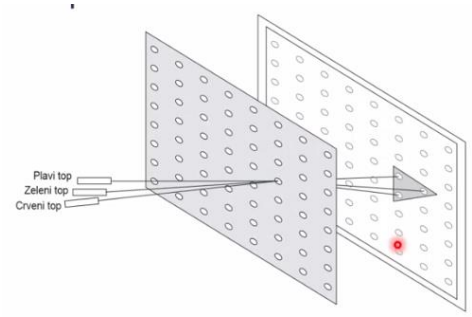
Odabir na koji način predstaviti podatke zavisi od potreba, ukoliko kreiramo neki model, pogodniji su nam vektorski podaci jer nam omogućavaju da zapamtimo taj model kao skup karakterističnih tačaka koje ga opisuju. Dok npr. kada napravimo digitalnu fotografiju fotoaparatom nama nije potreban skup objekata već prikaz slike skupom piksela različie boje koji će na kraju formirati čitavu sliku.

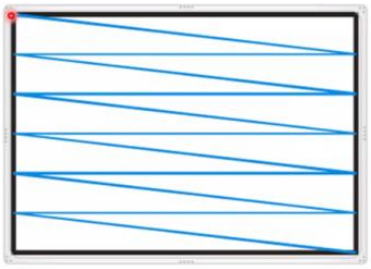
Kada se koriste vektorski podaci kvalitet slike se očuvava pri zumiranju i odzumiravanju jer postoje matematičke formule koje omogućavaju iscrtavanje objekta na osnovu karakterističnih tačakapod bilo kojim zumom. Kod rasterskih podataka to nije slučaj, rasterski podaci su matrica piksela, i kada se ta matrica zumira i odzumirava postoje tehnike mešanja boja piksela koje omogućavaju koliko-toliko glađi prelaz pri promeni nivoa zuma, ali se svakao gubi na kvalitetu slike jer je skup piksel informacija koje imamo fiksan.

8. Princip rada CRT ekrana

Ekran je staklena ploča premazana fosforom. Fosfor se koristi jer ima osobinu da kada biva pogođen elektronima i u trenutku predaje energije od elektrona fosforu on ima osobinu da svetli. Ta svetlost vremenom slabi pa ako želimo stalno osvetljenje treba stalno da gađamo fosfor elektronskim mlazom – 50-60 puta u sekundi da se ne bi videlo treperenje ekrana. Da bi odgovarajuća tačka na ekranu bila pogođena potrebno je da imamo i ostatak sistema. Na samom početku imamo jedan element koji služi za zagrevanje i on greje katodu (elektronski top). Kada se katoda zagreje ona počinje da emituje elektrone. Iza katode nalazi se kontrolna mreža koja kontroliše broj elektrona koji se šalju što će na kraju uticati na intenzitet osvetljenosti piksela. Zatim sledi sistem za fokusiranje koji služi da formira tanak mlaz elektrona. Takav tanak mlaz dolazi do dela ekrana koji sadrži kalemove za skretanje koji na bazi informacija koje dobijaju o tome koji piksel treba biti pogođen na ekranu skreću koliko je potrebno taj mlaz elektrona. Nakon skretanja, mlaz elektrona prolazi pored metalnih obloga sa visokim naponom koje služe da bi dovoljno ubrzale mlaz elektrona kako bi on pogodio odgovarajući piksel i preneo što veću količinu energije koja će na kraju izazvati da taj piksel svetli.

Prethodno opisani proces važi za monohromatske CRT ekrane, međutim i kod kolor ekrana je princip isti, s tim što za svaku komponentu boje postoji poseban elektronski top. Sam ekran sastoji se od niza tačaka koji predstavaljaju najmanje fizičke jedinice koje se mogu kontrolisati – dotovi. Dotovi ne moraju da se gađaju 1 na 1 sa pikselima na prikazu. Pravilo je da u maksimalnoh rezoluciji koju ekran može da podrži imamo slikanje 1 dot = 1 piksel, dok se pri manjim rezolucijama slika više dotova na 1 piksel. Kretanje mlaza elektrona koje emituje elektronski top je takvo da se slika na ekranu formira linija po linija, odozgo na dole, pri čemu te linije imaju određeni nagib i formiraju „cik-cak“ popunu ekrana. Od gornjeg levog ugla polazi top, ka desnoj ivici ide sa nagibom, kada dođe do desne ivice vraća se na levu ivicu u istom nivou i sve tako dok ne dođe do donjeg desnog ugla ekrana kada se top vraća na gornji levi ugao.





9. Načini iscrtavanja ekrana

Postoje dva načina iscrtavanja ekrana:

- **Bez preplitanja (non-interlaced)** – Postoji 1 frejm i on se iscrtava u svakom frejm periodu. Sve linije tog jednog frejma iscrtavaju se 50-60 puta u sekundi. Uglavnom se koristi danas.
- **Sa preplitanjem (interlaced)** – Postoje dva frejma, jedan sadrži parne, a jedan neparne linije. U jednoj poluperiodi frejma se iscrtava jedan frejm a u drugoj poluperiodi se iscrtava drugi frejm. Imamo istu frekvenciju iscrtavanja kao kod non-interlaced ali se iscrtava svaki put samo pola slike, što je našem oku prihvatljivo, ono ne vidi da nedostaju neke linije kada se frejmovi menjaju takvom brzinom. Ovaj metod je davao dosta dobar prikaz kod ekrana sa manjim frekvencijama rada jer je omogućio manje treperenje slike, iako se suštinski iscrtavala samo polovina slike, iscrtavanja su bila češća te treperenje manje.

10. Šta su rezolucija i aspekt slike?

Rezolucija (R) predstavlja ukupan broj piksela koji mogu biti prikazani na ekranu tj. U aktivnoj oblasti. R = AP * AL (broj aktivnih piksela * broj aktivnih linija) – ova dva broja se nikada ne množe pa da se da rezultat nego se da upravo u ovom zapisu množenja.

Aspekt slike (Aspect ratio – AR) je odnos horizontale i vertikale i dugo u istoriji je on bio 4:3, a sada je sve češće 16:9 (wide screen). AR = AP/AL (ukupan broj piksela u liniji sa ukupnim brojem linija)

11. Šta su dobre, a šta loše strane CRT tehnologije?

Dobre strane:

- Jeftina izrada
- Slika je jednako vidljiva pod svim uglovima
- Daje mogućnost prikaza kvalitetnih slika u punom rasponu boja na visokim rezolucijama

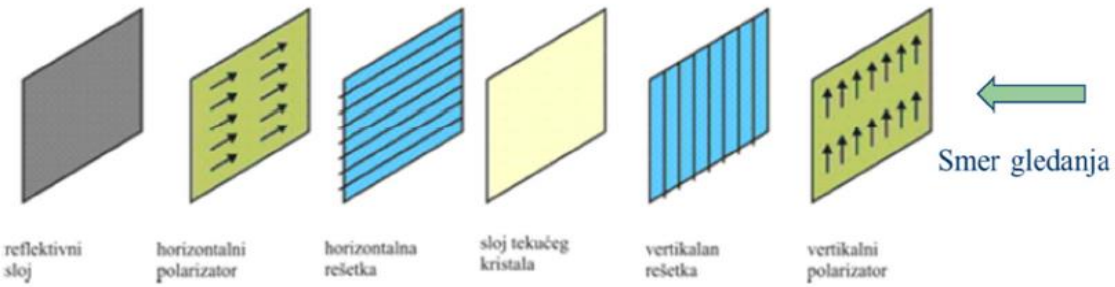
Loše strane:

- Glomazni
- Postoji deo sa visokim naponom pa je potrošnja električne energije velika
- Štetnost elektromagnetnog zračenja katodnih cevi
- Tehnologija prikaza zamara oči
- Osetljivost na spoljašnja elektromagnetna polja – ako približimo magnet nekom delu ekrana on može da se razmagnetiše i izgubi boje

12. LCD ekrani

LCD (Liquid Cristal Display) je tehnologija prikaznog uređaja koja koristi tečni kristal. Ova tehnologija postala je popularna za prenosne računare zbog znatno manjih dimenzija, težine i potrošnje energije od CRT uređaja. Tehnologija se bazira na šestoslojnoj strukturi:

1. Prednji sloj je **vertikalna polarizacijska ploča** – najbliži nama
2. Drugi sloj sadrži **vertikalnu rešetku** od tankih žica
3. Treći je **sloj tečnog kristala** debljine reda jednog mikrona – najbitniji sloj, molekuli kristala se nameštaju tako da ili polarizuju ili ne polarizuju svetlost koja dođe kroz horizontalnu rešetku. Ulazna svetlost je horizontalno polarizovna, a da bi izlazna svetlost bila vidljiva mora biti vertikalno polarizovana. Tako da ukoliko ovaj sloj okrene polarizaciju za 90 stepeni, takva svetlost će proći kroz vertikalnu rešetku i mi ćemo na vertikalnoj ploči da vidimo piksel. Ukoliko napravimo takvo elektromagnetno polje da se ne okrene polarizacija na mestu nekog piksela, ta svetlost neće proći kroz vertikalnu rešetku i mi ćemo taj piksel videti kao tamni piksel na ekranu. Piksel se dobija kao presek horizontalne i vertikalne rešetke. I ukoliko je vertikalna žica pod pozitivnim naponom, a horizontalna pod negativnim onda se tačka neće videti.
4. Četvrti sloj je **horizontalna rešetka** od tankih žica.
5. Peti sloj je **horizontalna polarizacijska ploča**
6. Poslednji, šesti sloj je **reflektor** – obezbeđuje osvetljenje ekrana. Ako je to neki obični reflektor onda je to LCD, ako koristi LED diode onda je to LED ekran itd.



13. Tehnologije za izradu video projektora

Postoje dve tehnologije za izradu projektora:

- **LCD** – tehnologija koja koristi tečne kristale. Slika se formaira pomoću tri odvojena TFT displeja, pri čemu je svaki od njih zadužen za jednu od boja – crvena, zelena plava. I ti ekrani, u kombinaciji sa optičkim elementima slažu RGB komponente slike u celinu. Takva celina se projektuje kroz objektiv na platno.
    - **Dobre strane** – potpuna zrelost tehnologije, vrlo kvalitetna reprodukcija boje i jeftina tehnologija izrade
    - **Loše strane** – veće dimenzije od projektora DLP tehnologije, nemogućnost obezbeđivanja visokog kontrasta pri prikazu slike i gubitak na kvalitetu slike posle 3-4 godine zbog slabljenja LCD ekrana
- **DLP (Difital Light Processing)** – potpuno digitalna tehnologija izrade video projektora. Njena suština je činjenica da na mestu svakog piksela ima jedno malo ogledalo koje je u obliku romba i ono može da rotira oko svoje duže dijagonale. Menjanjem položaja svakog od ovih ogledala kontroliše se količina svetla koja se usmerava u pravcu objektiva čime se kontroliše osvetljenost svakog od piksela. Može precizno da se kontroliše količina svetlosti koja će da se sprovede kroz objektiv na mestu svakog piksela. Boje se postižu korišćenjem rotirajućeg diska koji na sebi ima obojene filtere i koji se nalazi između čipa sa ogledalima i objektiva. Taj disk se okreće dosta brzo (50-100 puta u sekundi) čime dobijamo stabilnu sliku bez treperenja. Rotirajuči disk uglavnom ima 3 filtera (crveni, zeleni i plavi), ali može da ima i 6 (po 2 od svake od RGB boja), dodatnim filterima se postiže veći asortiman boja ali se blago smanjuje kontrast. Takođe postoji mogućnost da se na disk doda i jedan beli segment, čime se malo smanjuje kvalitet reprodukcije boja ali se povećava kontrast i kvalitet bele pozadine.
    - **Dobre strane** – bolji kontrast (ukoliko se koristi i beli segment) i značajno manje dimenzije
    - **Loše strane** – za nijansu veća cena, nešto veća buka koja se javlja kao posledica stalnog okretanja diska sa filterima

14. Tehnike prikazivanja i gledanja 3D videa

- **Naočare/kaciga sa posebnim ekranima za svako oko** – po jedan LCD ekran za svako oko. Naš mozak sam kreira 3D sliku na osnovu slika koje se serviraju svakom oku zasebno. Ovakve naočare/kacige su izuzetno složene i skupe, sam uređaj je veće mase i samo jedna osoba može da ih koristi u jednom trenutku.
- **Crveno-plave naočare** – Ispred jednog oka je crvena folija, a ispred drugog plava. Sam prikaz se sastoji od crvenih i plavih linija, pa se kroz crvenu foliju vide samo plave linije, a kroz plavu samo crvene linije. Na taj način se omogućava (odgovarajućim načinom iscrtavanja linija) kreiranje 3D slike. Ovaj princip je monohromatski.
- **Naočare sa polarizovanim staklima** – Ovakve naočare su lagane, ne opterećuju, podsećaju na bilo koje druge naočare. Stakla ovih naočara su polarizovana, ispred jednog oka je vertikalno polarizovano staklo, a ispred drugog, stako koje je horizonralno

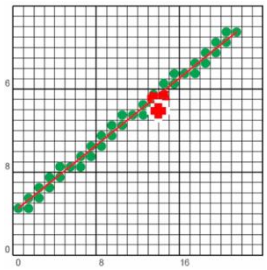
polarizovano. Koriste se dva projektora, jedan emituje vertikalno polarizovanu sliku, a drugi horizontalno polarizovanu sliku koja je pomerena. Kada se gleda kroz naočare, staklo sa vertikalnom polarizacijom propušta samo vertikalno polarizovanu sliku, a staklo sa horizontalnom polarizacijom samo horizontalno polarizovanu svetlost. Ljudski mozak onda sam kreira 3D sliku na osnovu slika koje dobija svako oko zasebno.

- **Active Shutter naočare** – Aktivne naočare posebno sinhronizovane sa ekranom. Ove naočare imaju LCD ekrane koji ne emituju sliku već naizmenično menjaju svoje stanje iz providnog u zatamljeno i moraju biti u stalnoj sinhronizaciji sa ekranom. Kada se prikazuje slika za levo oko naočare zatamne prikaz desnom oku i obrnuto. Ova zatamljenja i propuštanja se dešavaju velikom brzinom pa sam gledalac ne primećuje to dešavanje. Ove naočare su skupe, imaju LCD ekrane i dodatnu elektroniku za sinhronizaciju sa ekranom. Takođe, naočare se mogu koristiti samo zajedno sa ekranima koji moraju biti kvalitetni i imati mogućnost da menjaju slike velikom brzinom. Doživaljaj 3D prikaza je znatno bolji kod ovog modela nego kod polarizovanih naočara. Ali način rada ovih naočara podrazumeva stalno teperenje što dovodi do naprezanja očnog nerva i iako čovek ne vidi treperenje, ono mu može izazvati zdravstvene probleme – glavobolju, nagon za povraćanjem ... Takođe, ove naočare su dosta glomazne, tako da se ne mogu staviti preko naočara za vid, dok se polarizovane naočare mogu i tako nositi.
- **Specijalni TV ekrani koji ne zahtevaju nikakve naočare** – Ovo je najavljena tehnologija od strane mnogih proizvođača ali joj se i dalje ne znaju detalji. 3D sliku će biti mogue videti samo ako stojite pod određenim uglovima ispred ekrana, postoji 8 takvih uglova.

15. Algoritmi za skaniranje linije

Svaki algoritam može da se padne zasebno kao i da bude data konkretna duž sa konkretnim koordinatama i algoritam a da mi navedemo koje tačke, sa kojim koordinatama, će biti izabrane za iscrtavanje

- **Naivni algoritam (Brute-force)** – Pronalaze se svi pikseli koje seče data duž i svaki od njih se boji. To rezultuje u ne baš najboljem prikazu sa nejednakim brojem piksela po vrsti/koloni. Takođe se dosta procesorskog vremena potroši na provere preseka duži sa pikselima.



- **Nagibni algoritam** – Nastao kao poboljšanje i ubrzanje naivnog algoritma. Polazi od jednačine prave kojoj pripada ta duž, oblika y = m*x + b. Gde je m koeficijent pravca prave tj. Njen nagib. Za svako x od početnog do krajnjeg računamo vrednost y po jednačini prave.

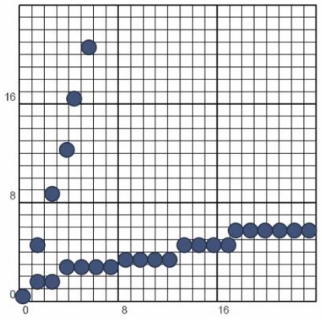
```
void NagibniAlgoritam (CDC* pDC, int x0,int y0,int
x1,int y1,COLORREF value)
{
  float dx, dy, m, y, b;
  int x;
  dx = x1 - x0;
  dy = y1 - y0;
  m  = dy/dx;
  b  = y1 - m*x1;

  for (x = x0; x <= x1; x++)
  {
      y = m*x + b;
      WritePixel(pDC, x, int(y+0.5), value);
  }
}
```

```
void WritePixel(CDC* pDC, int x, int y,
COLORREF value)
{
     pDC->SetPixel(x,y,value);
}
```

x0 < x1

Funkcija ima kao parametar pokazivaš na device context tj. Na uređaj na kojem će se izvršavati iscrtavanje, koordinate tačke početka i tačke kraja duži i boju kojom se iscrtava linija. U telu funkcije izračunavamo m i b i onda u petlji za svako x od x0 do x1 izračunavamo y po jednačini prave i pozivamo iscrtavanje dobijenog piksela. Koordinata y se zadaje kao int(y + 0.5), gde je y = izračunata vrednost pomoću x i jednačine prave. Na ovaj način radimo matematičko zaokruživanje na ceo piksel. int() je f-ja u C-u koja vrši odsecanje razlomljenog dela, zato dodajemo najpre 0.5 na y da bismo izvršili zaokruživanje svega što je ispod n.5 na n a svega preko n.5 na n+1. U samoj petlji u kojoj se računaju vrednosti piksela radi se sa realnim brojevima, što je sporije nego izračunavanje sa celobrojnim vrednostima. Takođe, daje loše rezultate za nagibe koji su preko 45 stepeni, tada algoritam daje mali broj tačaka jer je opseg po x mali



- **Inkrementalni algoritam** – Digitalni diferencijalni analizator (DDA), doneo je značajno poboljšanje u brzini. Eliminiše množenje definisanjem zavisnosti između y vrednosti u tekućem koraku i u prethodnom. Koristi se jednačina prave oblika y = m*x+b koja se izračuna na osnovu koordinata početne i krajnje tačke duži. Zatim se u jednačini izračunavanja y_{i+1} uvodi smena $x_{i+1} = x_i + dx$, a kako radimo sa pikselima, x se uvek menja za 1 tj. Dx je uvek = 1. Pa dobijamo zavisnost $y_{i+1} = y_i + m$.

```
void LineDraw(CDC* pDC, int x0, int y0, int x1, int y1,
COLORREF value)
{
     int x;
     double dy = y1 - y0;
     double dx = x1 - x0;
     double m = dy / dx;
     double y = y0;

     for (x = x0; x <= x1; x++)
     {
          WritePixel(pDC, x,int(y+0.5),value);
          y += m;
     }
}
```

$$m = \frac{dy}{dx} = \frac{y2 - y1}{x2 - x1}$$

$$y_i = mx_i + b$$

$$y_{i+1} = mx_{i+1} + b = m(x_i + dx) + b = mx_i + mdx + b = y_i + mdx$$

Ako je $dx = 1$, onda je:

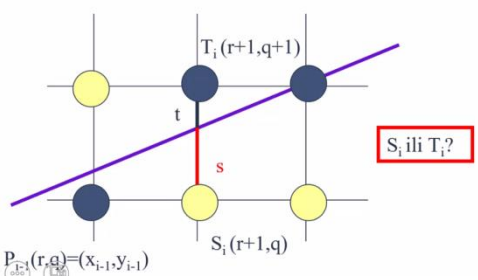
$$y_{i+1} = y_i + m$$

čime je eliminisano množenje.

- **Bresenhamov algoritam** – Tehnika se svodi na odlučivanje u svakom koraku kada se povećava x, da li treba povećati i y. (Pretpostavke u ovoj reprezentaciji algoritma su da je x0 < x1 I da je nagib linije u granicama 0 < m < 1).

  Kada god se traži Bresenham ide izvođenje i C kod ili primena na primeru. Treba da sami vodimo računa o tome da li se skalira po x ili po y!!!! (Svi algo u primerima su za x)

  Ako posmatramo piksele opšteg oblika P(x,y) koji imaju kokretne celobrojne koordinate i matematički tačnu duž, onda za svaku vrednost x duž prolazi između 2 piksela sa različitim vrednostima y. Treba odlučiti koji od ta dva piksela odabrati za reprezentaciju duži za dato x. Da bi se izv ršio odabir najpre se izračunava razdaljina tačke preseka matematički tačne duži sa linijom koja spaja ta dva piksela i svakim od piksela. Dužina koja je kraća pripada pikselu koji treba odabrati tako da reprezentuje duž za dato x. Nakon prethodnog koraka algoritma gde imamo izračunato da je $P_{i-1}(r,q) = (x_{i-1}, y_{i-1})$ onda možemo dva piksela između kojih se odlučuje da obeležimo sa $T_i(r+1, q+1)$ i $Si(r+1, q)$. Napre, da bismo izbegli izračunavanje sa slobodnim članom, mi ćemo translirati pravu u koordinatni početak i predstaviti je samo kao y=m*x. Sada treba izračunati razdaljine t i s. Razdaljina t se računa kao razlika razdaljine T_i od x ose, i razdaljina tačke preseka od x ose. Tačka preseka pripada matematičkoj duži pa se njena razdaljina od x ose može izračunati kao vrednost y u tački x po jednačini prave. Razdaljina s se računa kao razlika razdaljine tačke preseka od x ose i razdaljine piksela S_i od x ose. Zatim računamo razliku s – t i ukoliko je ona veća od 0 treba birati tačku T, a ukoliko je manja tačku S.

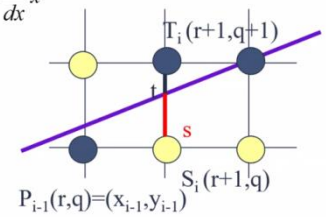


- Ukoliko transliramo liniju u koordinatni početak (za $T(-x_0,-y_0)$), onda je:

$$dx = x_1 - x_0 \qquad dy = y_1 - y_0 \qquad y = \frac{dy}{dx}x$$

$$s = \frac{dy}{dx}(r+1) - q$$

$$t = q + 1 - \frac{dy}{dx}(r+1)$$



$$s - t = 2\frac{dy}{dx}(r+1) - 2q - 1$$

$$dx(s-t) = 2(rdy - qdx) + 2dy - dx$$

d_i $\qquad r = x_{i-1}, q = y_{i-1}$

$$d_i = 2(rdy - qdx) + 2dy - dx$$

$$d_i = 2x_{i-1}dy - 2y_{i-1}dx + 2dy - dx$$

$$d_{i+1} = 2x_i dy - 2y_i dx + 2dy - dx$$

$$d_{i+1} - d_i = 2dy(x_i - x_{i-1}) - 2dx(y_i - y_{i-1}) \quad (x_i - x_{i-1} = 1)$$

$$d_{i+1} = d_i + 2dy - 2dx(y_i - y_{i-1})$$

Promenljiva odluke

d_i je promenljiva odluke $d_i = dx(s_i - t_i)$ tako da nam njen znak govori o tome da li je inkrementiran y u koraku I ili nije. Tj. Daje nam vrednost $y_i - y_{i-1}$

$$d_i = \begin{cases} \geq 0 & y_i = y_{i-1} + 1, \quad d_{i+1} = d_i + 2(dy - dx) \\ < 0 & y_i = y_{i-1}, \quad d_{i+1} = d_i + 2dy \end{cases}$$

Prateći ova pravila lako dolazimo do implementacije u jeziku C.

```
void BresenhamLine(int x0,int y0,int x1,int y1,
COLORREF value)
{
        int dx,dy,x,y,incr1,incr2,xend,d;
        dx = abs(x1 - x0);
        dy = abs(y1 - y0);
        d = 2*dy - dx;
        incr1 = 2*dy;
        incr2 = 2*(dy-dx);
        if (x0 > x1){
        x = x1;
        y = y1;
        xend = x0;
        }
        else {
                x = x0;
                y = y0;
                xend = x1;
        }
        WritePixel(pDC, x, y, value);
        while (x < xend) {
                x = x + 1;
                if (d < 0) d += incr1;
                else{
                        y = y + 1;
                        d += incr2;
                }
                WritePixel(pDC, x, y, value);
        }
}
```

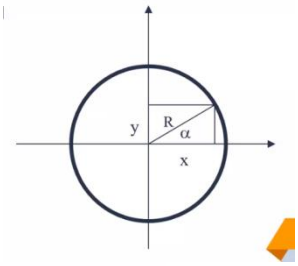
U glavnoj petlji imamo samo sabiranja. A množenja koja imamo na početku su množenja sa 2 koja se mogu realizovati šiftovanjem koje se smatra najbržom mogućom operacijom množenja. To sve upravo čini ovaj algoritam najbržim algoritmom za rasterizaciju linije.

Postoji i varijacija ovog algoritma sa istom kompleksnošću koja se naziva Midpoint algoritam.

16. Algoritmi za skaniranje kruga

Uvedena je pretpostavka da se crta kruga sa centrom u koordinatnom početku, kako bi nam bilo lakše.

- **Trigonometrijski algoritam** – Trivijalni algoritam. Radi na osnovu parametarskih jednačina kružnice. Ukoliko znamo poluprečnik kružnice – R, možemo da se krećemo po kružnici za vrednost ugla od 0 do 2*pi i da izračunavamo vrednosti x i y, x = R*cosα i y=R*sinα. Kod ovakvih algoritama izuzetno je bitan način na koji biramo korak promene ugla. Ukoliko je taj korak fiksan onda dobijamo različitu preciznost crtanja za krugove sa malim, odnosno velikim poluprečnikom. Kako bi se održala preciznost crtanja, korak promene ugla mora biti manji za uglove sa većim R, odnosno veći za krugove sa manjim R.

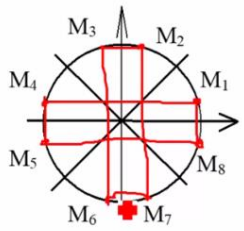


$x = R\cos\alpha,$
$y = R\sin\alpha,$
za $\alpha \in (0, 2\pi)$

```
void TrigCircle(CDC* pDC, int r, COLORREF value)
{
      float alfa, dvapi = 6.283185;
      float step = dvapi / (7*r);
      int x, y;
      for (alfa = 0; alfa < dvapi; alfa += step)
      {
            x = int(r*cos(alfa) + 0.5);
            y = int(r*sin(alfa) + 0.5);
            WritePixel(pDC, x, y, value);
      }
}
```

Korak se računa kao (2*pi)/(7*R), odakle se vidi da je korak obrnuto proporcijalan poluprečniku kružnice. Ovaj algoritam je jednostavan za implementaciju ali koristi računicu sa realnim brojevima, množenje i funkcije sinusa i kosinusa koje se izvršavaju u svakom koraku pa je algoritam dosta spor.

Osnovni trigonometrijski algoritam se može ubrzati. Možemo da primetimo da je krug visoko simetrična figura, pa izračunavanjem x i y za jednu tačku, zapravo imamo potrebne vrednosti za 8 tačaka na krugu. Što praktično znači da je potrebno da izračunamo tačke za svega osminu kruga koje će se iskoristiti za predstavljanje tačaka preostalih 7 osmina. Ugao, dakle, može da se kreće od 0 do pi/4 čime se drastično povećava brzina izvršenja algoritma.

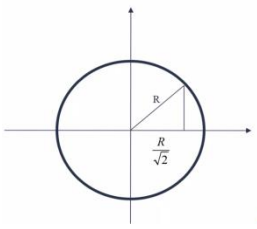


M_1 = (x,y)	M_5 = (-x,-y)
M_2 = (y,x)	M_6 = (-y,-x)
M_3 = (-y,x)	M_7 = (y,-x)
M_4 = (-x,y)	M_8 = (x,-y)

```
void SimetricCircle(CDC* pDC, int r, COLORREF value)
{
      float alfa, dvapi = 6.283185, pi4 = 0.785398;
      float step = dvapi / (7 * r);
      int x, y;
      for (alfa = 0; alfa < pi4; alfa += step)
      {
            x = int(r*cos(alfa) + 0.5);
            y = int(r*sin(alfa) + 0.5);
            WritePixel8(pDC, x, y, value);
      }
}
```

- **Polinomni algoritam** – Kreće od polinomne jednačine kružnice. Ako se krećemo po x osi, za svako x možemo da računamo y po polinomnoj formuli. X kreće od koordinata cetra kružnice, i ide do vrednosti R/√2. Izračunavanjem y za taj deo kruga dobijamo 1/8 kružnice (M2 po slici iznad) i na osnovu pravila simetrije možemo odrediti i ostatak kružnice.

$$x^2 + y^2 = R^2$$
$$y = \sqrt{R^2 - x^2}$$




```
void PolinCircle(CDC* pDC, int r,
COLORREF value)
{
     int x, y, xend;
     xend = (int)(r/sqrt(2)+0.5);
     for(x = 0; x <= xend; x++)
     {
          y=(int)(sqrt(r*r-x*x)+ 0.5);
          WritePixel8(pDC, x, y, value);
     }
}
```

Algoritam je nešto brži nego trigonometrijski, ali i dalje imamo operacije sa realnim brojevima, množenje i kvadratni koren.

- **Bresenhamov algoritam** –

17. Koje su to perceptivne veličine kojima se opisuje boja?

Boja se opisuje pomoću 4 veličine – nijansa, zasićenost, svetlina i sjajnost.

- **Nijansa** – komponenta koja definiše samu boju, pravi razliku između boja.
- **Zasićenost** – predstavlja meru koliko je boja daleko od sive boje istog intenziteta. Na primer, crvena boja je visoko zasićena, dok je roze boja relaivno nezasićena. Generalno su sve jarke boje zasićene, a sve pastelne nezasićene.
- **Svetlina** – opisuje intenzitet boje, tj. Uključuje monohromatsku notaciju za osećaj intenziteta reflektujućeg objekta.
- **Sjajnost** – koristi se umesto svetline da bi opisala intenzitet boje za koju se smatra da emituje svetlost.

18. Veličine koje definiše kolorimetrija

Kolorimetrija je grana nauke koja se bavi bojama i koja definiše tri osnovna pojma – dominantna talasna dužina, čistoća eksitacije i svetlenje.

- **Dominantna talasna dužina** – predstavlja talasnu dužinu boje koju vidimo kada gledamo u svetlo i odgovara komponenti *nijanse.*
- **Čistoća eksitacije** – odgovara *zasićenju* i proporcionalna je čistoći svetla dominantne talasne dužine i belom svetlu potrebnom za definisanje konkretne boje.
- **Svetlenje** – odgovara intenzitetu svetlosti.

19. Šta je model boja?

Model boja je specifikacija 3D koordinatnog sistema boja. Cilj modela je da omogući pogodnu specifikaciju boja unutar neke skale boja. Navođenjem s koordinate jednoznačno se određuje boja koja odgovara tim koordinatama.

20. Nabrojati modele boja

Može da bude i pitanje da se objasni neki od ovih modela boja i da se neka RGB boja predstavi u nekom od ovih formata

- **CIE model –** najstariji model. Sastoji se od tri komponente – X, Y i Z. Ove komponente predstavljaju hipotetičke boje koje u stvarnosti ne postoje ali se njihovim mešanjem može dobiti bilo koja realna boja. Ovaj model uvodi i pojam hromatičnosti, svaka od 3 komponente ima svoju hromatičnost x = X / (X + Y + Z), y = (X + Y + Z) i z = (X + Y + Z). Hromatičnost zavisi samo od dominantne talasne dužine, a ne i od količine energije svetla. Dijagram hromatičnosti predstavlja projekciju 3D modela boje na XY ravan. Nijedan uređaj ne može da prikaže sve boje sa dijagrama hromatičnosti, već se definiše neki njegov deo koji može biti prikazan i taj deo naziva se GAMA tog uređaja. Model je manje prirodan od RGB modela jer se RGB bazira na realnim bojama, a CIE na hipotetičkim. Ali je ovaj model zgodan za transformaciju boje iz jednog modela u drugi –

ako ne znamo direktnu formuli za transformaciju boje iz jednog modela u drugi, uvek možemo transformaciju učiniti u tri koraka, posredovanjem CIE modela.

- **RGB model** – Trodimenzionalan model, zasnovan na Dekartovom koordinatnom sistemu, pri čemu su sada X, Y i Z komponente zapravo R, G i B komponente. Opseg boja varira, ukoliko koristimo normalizovan model onda svaka od ovih boja ima opseg 0 do 1. Za potrebe predstavljanja u računaru često se koristi opseg 0 – 255. Vrlo često se u grafici prilikom korišćenja ovog modela definiše i četvrta komponenta – alfa. Ova komponenta opisuje koliko je prozirna neka boja, tj. Koliko se providi površina obojena tom bojom. RGB model pripada grupi aditivnih modela – boje se dobijaju dodavanjem crnoj boji, a mešanjem crvene, zelene i žute dobija se bela boja. Svaka boja definiše se procentom prisustva osnovnih boja u ovoj boji.
- **CMY (Cian – Magenta – Yellow) model –** Tri osnovne boje po ovom modelu su komplementi crvene, zelene i plave. Pripada grupi subtraktivnih modela – boje se dobijaju tako što se oduzimaju od bele boje, a mešanjem svih osnovni boja dobija se crna boja. Zgodan je za uređaje koji štampaju po beloj podozi. Ovaj model je komplementaran RGB modelu pa je konverzija iz jednog u drugi laka. Za opseg vrednosti 0-1 potrebno je samo od 1 oduzeti R, G i B vrednosti i dobiće se C, M i Y vrednosti. Na sličan način može se obaviti i konverzija u suprotnom smeru.
- **CMYK model** – Model koji je nastao kao nadogradnja CMY modela. Uvodi u crnu boju kao zasebnu komponentu. Nastao je usled razvoja štampača, pri čemu se došlo do zaključka da je bolje imati i crnu boju zasebno jer se ona najčešće štampa i nije efikasno stalno trošiti ostale tri komponente za mešanje i dobijanje crne boje.
- **YIQ model** – Koristi se kod crno-belih monitora. Obezbeđuje da se sve različite boje slikaju u različite nijanse sive čime se omogućuje adekvatno praćenje sadržaja.
- **HSV (Hue – Saturation – Value) model** – Svi prethodni modeli su HW orijentisani, dok su HSV i HLS modeli korisnički orijentisani i namenjeni umetnicima. Koordinatni sistem kod ovog modela je cilindričan, tj. Osnova je heksakonus na čijim su ivicama raspoređene osnovne boje – crvena, žuta, zelena, cijan, plava i magenta, dok je u samom centru bela boja. Osnovne boje su raspoređene na po 120 stepeni, a komplementarne boje nalaze se u razmaku od 180 stepeni. Ceo model predstavlja geometrijsko telo koje nastaje na osnovi ovako definisanog šestougaonika. Na samom vrhu heksakonusa nalazi se crna boja. *Hue* se specificira kao ugao – crvena je na uglu 0, žuta na uglu od 60 stepeni, ... Saturacija se definiše na x osi, a *value* po y osi.
- **HLS (Hue – Lightness – Saturation) model** – Sličan prethodnom modelu, s tim što se centar geometrijskog tela „ivlači“ u dva različita smera – na vrhu jednog heksakonusa je bela boja, a na vrhu drugog crna. Na ovaj način se povećava opseg za svetlinu u odnosu na HSV model.

21. Model osvetljenja i model senčenja

Model senčenja je širi okvir i on koristi model osvetljenja.

22. Nabrojati modele osvetljenja

- **Samo-osvetljenost** – njjednostavniji model osvetljenja. Za svaki objekat, svakoj tački tog objekta pridružen je isti intenzitet svetlosti. Ovo nije realističan model.
- **Ambijentalno svetlo** – Model koji podrazumeva difuziono svetlo bez usmerenog izvora. Ovakvo svetlo postoji i u realnom svetu i proizvod je višestrukog odbijanja svetlosti od raznih površina prisutnih u sceni. Ambijentalno osvetljenje je neophodno da bi modelovalo svu svetlost koja ne dolazi iz nekog konkretnog izvora, ali nije dovoljno da samostalno učini scenu realističnom, za takav efekat mora se kombinovati sa još nekim tipom svetla.
- **Tačkasti izvori svetla** – Podrazumeva se da je izvor svetlosti tačkast i da ravnomerno širi zrake svetlosti u svim smerovima. Kod ovih svetala potrebno je definisati difuznu refleksiju. Kako će izgledati površine kada su osvetljene tačkastim izvorom zavisiće od ugla između pravca svetla i pravca normale na tu površinu.
- **Direkciono svetlo** – Može se svesti na pojam tačkastog izvora svetla ukoliko taj izvor dovoljno udaljimo od svih objekata. Tada se može smatrati da su zraci koji padaju na cenu konstantni za sve objekte na sceni tj. Padaju na sve objekte pod istim uglom. Primer takvog svetla u realnosti je Sunčevo svetlo.
- **Spot svetlo** – svetlo koje se emituje iz nekog izvora u vrlo uskom ugaonom opsegu i samo jedan deo scene je osvetljen njime u svakom trenutku.
- **Prošireni izvori svetla** – Svi prethodno pomenuti modeli odnose se na tačkaste izvore svetlosti. Smatra se da je u pitanju tačkasti izvor koji može imati neku svoju realnu poziciju, ili može biti dovoljno udaljen da možemo da ga posmatramo kao direkciono svetlo. U realnosti izvori svetlosti nisu tačkasti, svetlost dolazi sa nekog objekta koji ima površinu koja emituje svetlost. Ovakva svetlost modeluje se proširenim izvorima.

23. Rendering

Rendering je postupak određivanja odgovarauće boje piksela koji je pridružen nekom objektu u sceni. Zavisi od geometrije objekata koji su osvetljeni, od geometrije, tipa, pozicije i boje svetlosnog izvora, od pozicije posmatrača, od materijala od kojih je napravljen objekat, ... Možw predstavljati izuzetno zahtevan posao da bi se uzeli svi pomenuti faktori u obzir.

24. Modeli senčenja

Modeli senčenja dele se u dve velike grupe – modele sa lokalnim pristupom i modele sa globalnim pristupom. Modeli sa lokalnim pristupom rade tako da se razmatra samo osvetljenje od lokalnih izvora svetlosti i ne uzima se u obzir refleksija od ostalih objekata u sceni. Dok se kod globalnog pristupa u obzir uzima I refleksija od ostalih objekata u sceni.

25. Modeli senčenja sa lokalnim pristupom

- **Konstantno senčenje** – kod ovog modela sve tačke površine unutar jednog poligpna imaju isti intenzitet. Ovde nemaju svi objekti niti čak svi delovi objekta isti intenzitet (kao kod samo-osvetljenosti), već sve tačke jednog poligona imaju isti intenzitet. U tačkama

koje pripadaju većem broju poligona istovremeno normala se računa kao aritmetička sredina normala u toj tački za sve poligone kojima tačka pripada. Prednost ovog modela je ta što nema kompleksnih izračunavanja, ali je nedostatak prevelika istaknutost ivica.

- **Gouraud-ovo senčenje** – Ovaj model senčenja je kompleksniji od prethodnog, ali ne isuviše kompleksan tako da se i dalje može koristiti za senčenje u realnom vremenu. Kod ovog modela se eliminiše glavni nedostatak prethodnog modela – vidljivost ivica. Ovo se postiže tako što se radi interpolacija boja i intenziteta osvetljenosti duž svakog poligona koji se nalazi na 3D površini. Ne računaju se normale na celu površveć na svako teme u toj površi zasebno. Zatim se računaju boje i intenziteti za sva temena poligona, zatim se vrši interpolacija duž ivica poligona, nakon čega se vrši interpolacija i po svakoj sken liniji. Rezultat ovakvog izračunavanja su glatki prelazi.
- **Phong-ovo senčenje** – Polazi se od toga da se nalazi normala za svako teme poligona koji posmatramo. Zatim se radi interpolacija normala na svaki piksel duž ivice, a isti proces se ponavlja i duž svake sken linije. Zatim se na osnovu ovako dobijenih normala vrši izračunavanje boje i intenziteta osvetljenosti svakog piksela. Ovo daje znatno tačnije rezultate od prethodnih modela ali je i računca dosta kompleksna pa ovaj model nije pogodan za senčenje u realnom vremenu.

26. Modeli senčenja sa globalnim pristupom

- **Metod praćenja zraka** – Ovaj metod uzima u obzir i senke i refleksiju, odbijanje zraka, refrakciju, prelemanje zraka kroz transparentne objekte. Uzima u obzir i ostale objekte na sceni i to da li oni utiču na piksel koji trenutno izračunavate. Podrazumeva isključivo tačkaste izvore svetlosti i računanje svakog piksela zavisi od položaja posmatrača zato što se zrak emituje iz oka posmatrača.
- **Metod isejavanja** – Uzima u obzir sve što uzima i metod praćenja zraka ali je njegova prednost što svetlosni izvori mogu da budu proizvoljnih dimenzija i geometrija, čime se znatno usložnjava izračunavanje ali se dobija na kvalitetu. Kod ovog metoda, izračunavanje ne zavisi od položaja posmatrača.
- **Metod praćenja putanje** – Najbolji metod ali i najzahtevniji u računskom smislu. Koristi Monte-Carlo metodu za upravljanje geometrijom, efleksijom i osveljajem.

27. Modeli senke

- **Lažne senke** – Senke koje ne predstavljaju realnu senku nekog objekta već se koriste kako bi nam dale neke dodatne informacije o relativnom položaju objekata na sceni. Prednost ovog modela je to što se senka može brzo generisati jer ima jednostavan oblik.
- **Projektovanje senke** – Daje realnu senku objekta tako što se odradi projekcija objekta iz pozicije izvora svetla na ravan na koju senka pada. Prednost je što se dobija realna senka, ali je nedostatak to što projekcija mora da se računa svaki put kada se objekat pomeri.
- **Mape senki** – Model koji se koristi za senke statičnih objekata. Odredi se senka objekta projekcijom i zatim se takva senka zapamti u teksturu koja se postavi ispod predmeta

tako da on konstantno ima senku bez dodatnih izračunavanja. Prednost ovog pristupa je što se smanjuje potreba za izračunavanjem projekcija, a nedostatak je to što ako se pomeri izvor svetlosti biće potrebno promeniti i senku.

- **Zapremina senke** – Model koji uzima u obzir koliko objekata zaklanja svetlost pre nego što se dođe do računanja senke za dati objekat. Čitav prostor se deli u zapremine za koje se vodi računa n akojoj su dubini, odnosno koliko objekata zaklanja svetlost koja dolazi do te zapremine. Onda objekti koji se nalaze u zapreminama sa većom dubinom moraju biti tamniji od objekata u zapremini sa manjom dubinom. Ovaj model utiče i na kreiranje senke i na objekte koji se iscrtavaju u oblasti obuhvaćenoj senkom.

28. Po čemu se razlikuju modeli za predstavljanje poligonalnih mreža

- Po potrebnom memorisjkom prostoru za smeštanje modela
- Po lakoći identifikovanja ivica koje su susedne u nekom temenu
- Po lakoći identifikacije poligona sa zajedničkom ivicom
- Po lakoći prikaza modela
- Po lakoći identifikacije grešaka u modelu

29. Koji se modeli koriste za predstavljanje poligonalne mreže?

- **Eksplicitna lista temena** – navode se tačke koje se memorišu prema redosledu obilaska. Ovaj model ni na koji način nije struktuiran i nije efikasan jer dolazi do ponavljanja temena, pa u slučaju promena, treba izvršiti istu za svako pojavljivanje. U ovom modelu nije jednostavno pronaći zajedničke i susedne ivice.
- **Lista poligona** – U memoriji se svaka tačka pamti samo jednom. Kada se onda definiše polygon koji sadrži neke tačke onda se koriste pokazivači na te tačke. Na ovaj način se razdvaja geometrija od topologije. Pri čemu geometrija određuje lokaciju temena, a topologija određuje organizaciju temena i ivica u poligone. Tako da ako se nekada menja položaj neke tačke (geometrija), to neće uticati na to kom poligonu ta tačka pripada (topologija).
- **Eksplicitna lista ivica** – Rešava probleme višestrukog iscrtavanja zajedničkih ivica kao i otežanog pronalaženja zajedničkih ivica, što je bio problem kod prethodna dva modela. Pored toga što se čuva jedinsvena lista tačaka, čuva se i jedinstvena lista ivica – svaka ivica se pamti samo jednom. Na ovaj način se omogućuje da se svaka ivica iscrta samo jednom ali je i dalje teško pronaći zajednička temena i ivice koje su susedne u nekom temenu.

30. Interpolacija i aproksimacija

Interpolacija se koristi kada želimo rekonstruisati originalnu krivu koja mora proći kroz zadate tačke. **Aproksimacija** se koristi kada se želi postići estetski efekat tako što se neka kriva aproksimira pravilnijom krivom (sa blažim promenama) tako da ona prolazi u blizini zadatih tačaka, a ne kroz njih.

31. Osnovni principi prikaza krivih

- **Kontrolne tačke** – Svaka kriva mora da ima određeni skup kontrolnih tačaka kako bi se ta kriva mogla kontrolisati (bez obzira na to da li je u pitanju interpolacija ili aproksimacija). Pomeranjem kontrolnih tačaka utiče se na izgled krive.
- **Višeznačne vrednosti** – krive u opštem slučaju ne moraju da imaju jednoznačne vrednosti (za jedno x, jedno y)
- **Nezavisnost od koordinatnog sistema** – oblik krive mora biti nezavisan od koordinatnog sistema. Kriva ima svoj oblik u prostor, a oko nje se onda može praviti koorinatni sistem kakav se želi.
- **Globalna i lokalna kontrola** – krive se mogu modelirati tako da imaju globalnu kontrolu – kada se promeni jedna kontrolna tačka menja se oblik čitave krive, ili lokalnu kontrolu – ako se promeni jedna kontrolna tačka menja se samo deo krive u okolini te tačke.
- **Varijacija između kontrolnih tačaka** – algoritam za crtanje krive mora da obezbedi da izračunata kriva nema više prevojnih tačaka od izvorne krive.
- **Titranje krive** – treba ga smanjiti.
- **Mogućnost dodavanja i brisanja kontrolnih tačaka** – algoritam ovo mora da dozvoli. Tamo gde je potrebna finija kontrola krive biće više kontrolnih tačaka, gde nije, biće ih manje.
- **Granični uslovi** – za krive koje se nastavaljaju na neke druge moguće je definisati granične uslove pomoću kojih se vrši kontrola reda neprekidnosti. Neprekidnost nultog reda podrazumeva da nova kriva počinje u tački u kojoj se stara završava. Neprekidnost prvog reda podrazumeva da dva segmenta krive u tački spajanja imaju istu tangent. Što se svodi na to da prvi izvodi u toj tački budu jednaki. Neprekidnost drugog reda podrazumeva da u tački spajanja rezultujuća kriva ima jedinstvenu krivinu, tj. Da segmenti krive imaju istu oskulatornu kružnicu, odnosno da su i prvi izvodi u toj tački jednaki i drugi izvodi jednaki. Neprekidnost trećeg reda podrazumeva da se u tački spoja održava kontinuitet u izvijanju, što se svodi na to da treći izvodi krivih u toj tački budu jednaki. Pored ovih tipova kontinuiteta, postoje I takozvani G kontinuiteti koji ne podrazumevaju jednakosti odgovarajućih izvoda već njihovu proporcionalnost.

32. Analitičko predstavljanje krivih *

Može da bude pitanje da se da bilo koja od ovih reprezentacija neke krive (prave) koja prolazi kroz zadate tačke.

Sve krive mogu da se prestave eksplicitno, implicitno i parametarski.

- **Eksplicitni oblik** – zadaje se u formatu y = f(x)
- **Implicitni oblik** – zadaje se u formatu f(x,y) = 0
- **Parametarski oblik** – x = f(t) i y=f(t), t obično ide 0-1. Tačka s emože redstaviti i u vektorskom obliku – kao vektor koji polazi iz koordinatnog početka i završava se u toj tački.

33. Prednosti parametarskog predstavljanja krivih

- Mogućnost predstavljanja krivih koje nemaju poznatu matematičku definiciju
- Sve koordinate se tretiraju na isti način

- Mogućnost izračunavanja višeznačnih vrednosti
- Razlika pri generisanju krive u 3D, u odnosu na 2D se svodi samo na dodatno izračunavanje z(t).

34. Najpoznatije krive koje se koriste u grafici

- **Bezierove krive** – Generišu se za skup kontrolnih tačaka, prolaze kroz početnu i krajnju tačku, a pored ostalih kontrolnih tačaka samo prolaze, što znači da se radi o aproksimaciji. Primer Bezierovog polinoma za n=3:

  $$Q(t) = (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t)P_3 + t^3 P_4$$

  Karakteristike Bezierove krive:

  - Postoji tzv. *Konveksna ljuska* koju obrazuju kontrolne tačke. Kada spojimo sve kontrolne tačke, one obrazuju neku konveksnu oblast u okviru koje se nalazi cela kriva.
  - Kriva nema više krivina od kontrolnog poligona. Broj preseka ravni sa krivom je uvek veći ili jednak od broja preseka ravni i kontrolnog poligona.
  - Nije ispunjena lokalna kontrola – ukoliko pomerimo neku kontrolnu tačku, Bezierova kriva će biti globalno promenjena.
  - Broj kontrolnih tačaka je u direktnoj vezi sa stepenom krive.
  - Nezavisne su od transformacija. Ukoliko se vrši translacija, rotacija ili skaliranje, Bezierova kriva ne menja svoj oblik.
  - Simetričnom zamenom kontrolnih tačaka ne menja se oblik Bezierove krive
- **B-splajnovi** – Generiše se na osnovu kontrolnih tačaka. U zavisnosti od toga da li su t intervali ravnomerno raspoređeni, odnosno da li su kontrolne tačke ravnomerno raspoređene po x osi, dele se na uniformne i neuniformne B-splajnove.

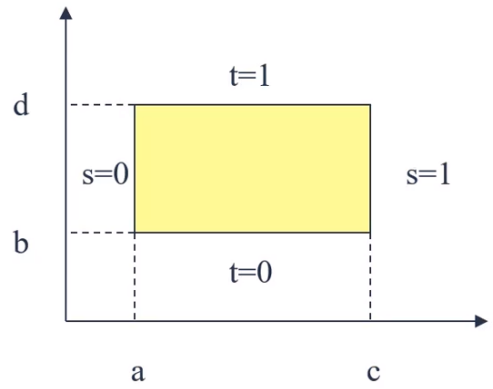
  Karakteristike B-splajnova:

  - Kriva leži unutrar konveksne ljuske
  - I svaki segment krive nalazi se unutar konveksne ljuske koju definišu kontrolne tačke za taj segment.
  - Postoji lokalna kontrola – pomeranje jedne kontrolne tačke utiče najviše na dva susedna segmenta na toj kontrolnoj tački
- **NURBS krive** – Non Uniform Rational B-Spline – Izvedena je iz B-splajna sa razlikom da svaka kontrolna tačka sada ima svoju težinu. To znači da nemaju sve kontrolne tačke isti uticaj na izgled krive – tačke sa većom težinom imaju veći uticaj.

35. Analitičko predstavljanje površi

- **Eksplicitni oblik** – $z = f(x,y)$
- **Implicitni oblik** – $f(x,y,z) = 0$
- **Parametarski oblik** – sve koordinate parametarski zavise od dva paramtra – s i t, koji se kreću u opsegu 0-1. $x = f(s,t)$, $y = g(s,t)$, $z = h(s,t)$

36. Parametarski oblik pravougaonika



$$x = (c-a)\cdot s + a$$
$$y = (d-b)\cdot t + b$$

$$s,t \in [0,1]$$

37. Najpoznatije površi koje se koriste u grafici

- **Bezierove površi** – Postoje kontrolne tačke koje utiču na površinu. Ne postoji okalna kontrola.
- **B-splajn površi** – Postoje kontrolne tačke. Ista podela na uniformne i neuniformne kao kod krivih. Imaju lokalnu kontrolu.
- **NURBS površi** – Izvedene iz B-splajn površi. Imaju kontrolne tačke i lokalnu kontrolu. Uvode težine za svaku kontrolnu tačku – nemaju sve kontrolne tačke isti uticaj na izgled površi.