**1. Windows GDI- opis i princip rada**

GDI (*Graphics Device Interface*) je 2D grafički API koji Microsoft koristi u svojim operativnim sistemima. GDI enkapsulira svoje funkcije i grafičke objekte u MFC klase (CDC - *Device Context*, *CBrush* - četka, *CPen* - olovka, *CBitmap* - bitmapa...). Mesto GDI-ja u *Windows* operativnom sistemu dato je na slici pored.

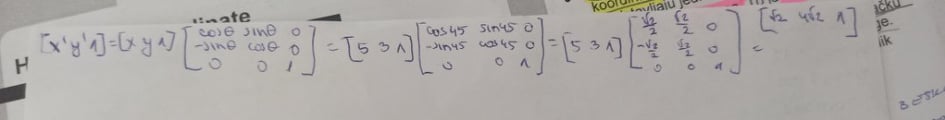
**Potencijalno podpitanje:**

**-Device Context.**

*Device Context* je apstraktni (virtuelni) uređaj. Kada se programira, pristupa se ovom uređaju i na taj način se ne vodi računa direktno o hardveru i njegovim karakteristikama. O karakteristikama hardvera brigu vode drajveri uređaja.

**3. Izračunati matricu za 2D rotaciju za ugao od 45 ° oko tačke P(5,3).**

*P '* =*P*⋅*R* ⟶[ *x ' y ' 1* ]=[ *x y 1* ] [ cos θ sin θ]



**4. Navesti modele senki koji se koriste u rac. grafici. Detaljno objasniti model koji koristi lažne senke. Šta su dobre a šta loše strane ovog modela?**

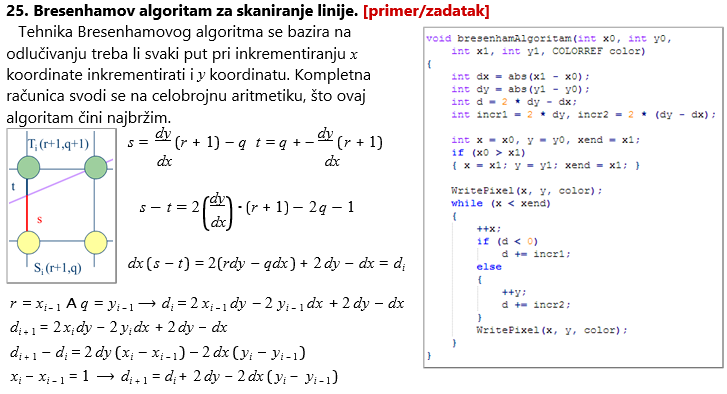
Od algoritama za određivanje senki postoje:

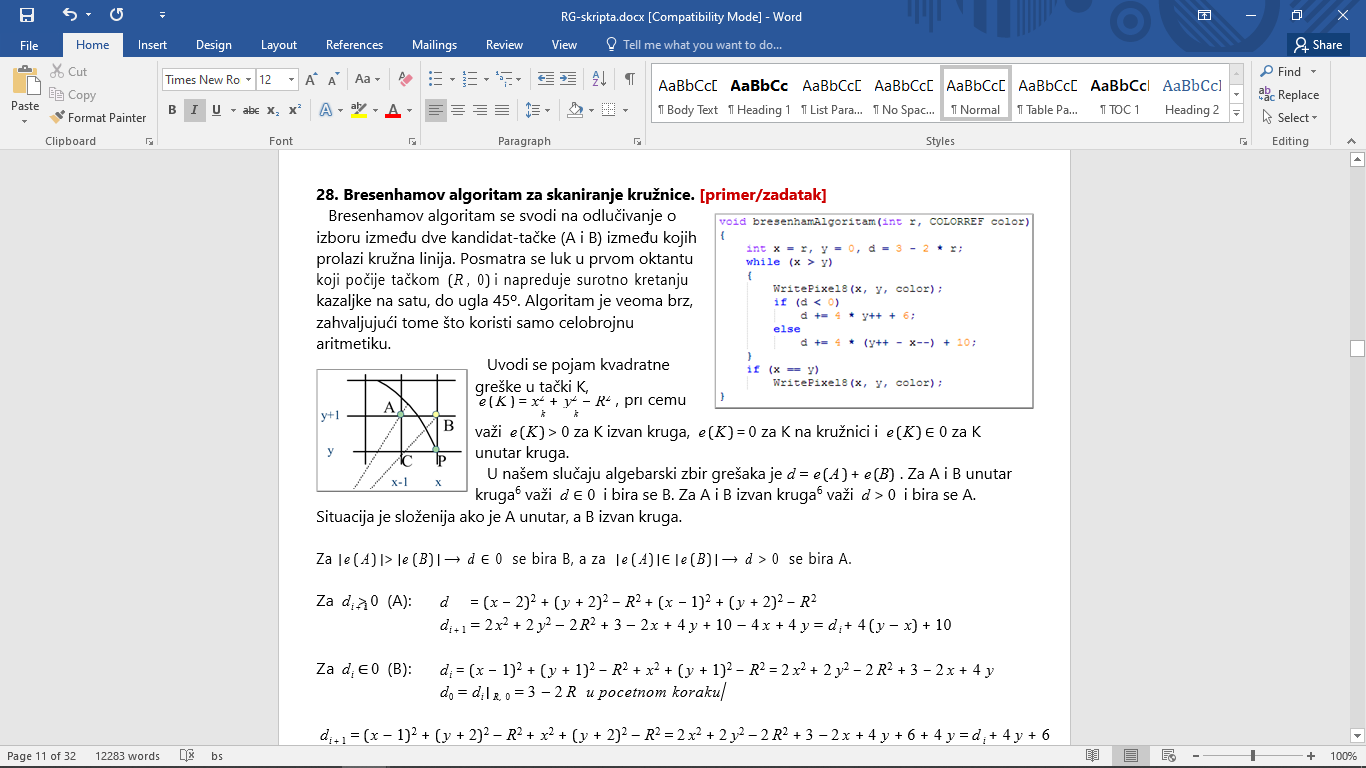
* 1. **Lažne senke:** Kao senke se koriste jednostavni objekti (poligoni pored/ispod objekta).
  2. **Projektovane senke**: Senke su projekcije objekata iz pozicije izvora svetlosti.
  3. **Mape senki:** Vrše se dva renderinga - prvi se vrši iz ugla izvora svetlosti, pri čemu se pamti dubina svake površine koju svetlo „vidi“, a drugi renderuje pogled posmatrača, uz poređenje svake tačke sa zapamćenom dubinom (mapom senki), radi utvrđivanja da li je površina osenčena ili ne.
  4. **Zapremina senke:** Za svaki objekat se određuje poluotvorena zapremina senke koja ima oblik najsličniji obliku zarubljene piramide. Ulazak zraka u zapreminu svake senke povećava dubinu senke, dok je izlazak iz zapremine smanjuje.

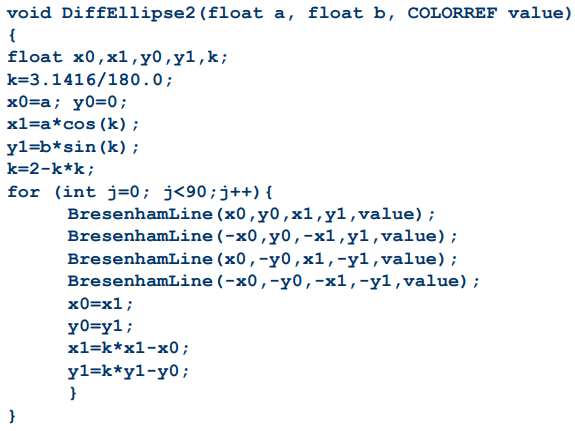
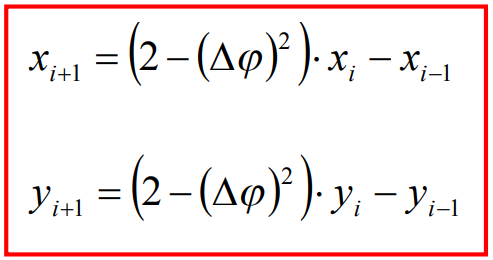
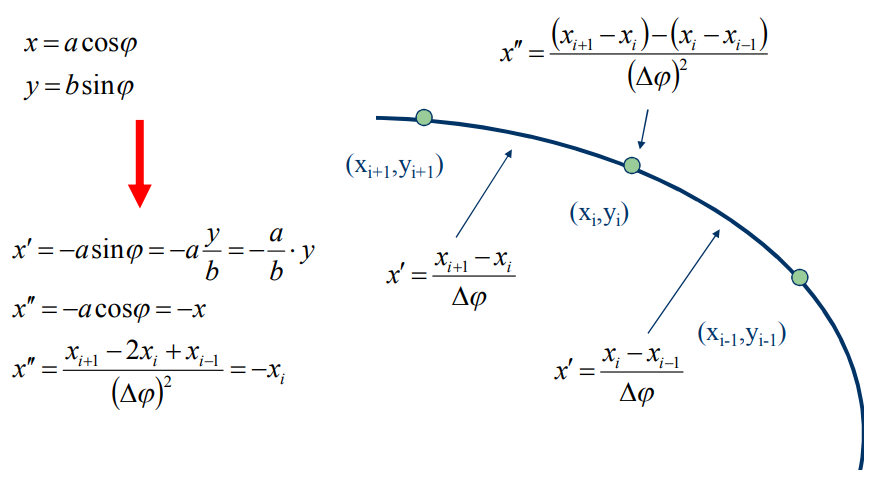
**5. Navesti modele boja koji se koriste u rac.gr. Koje vrednosti ima boja u CMY, ako je u RGB(235, 148, 228)? Vrednosti po obe su 0-255**

Model boja je specifikacija 3D koordinatnog sistema boja, čiji je cilj da omogući pogodnu specifikaciju boja unutar neke skale boja. Skala koja je od interesa ovde je skala za kolor CRT monitor.

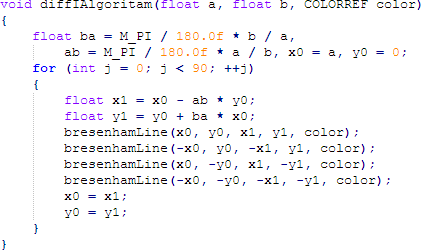
Postoji više modela boja: CIE, EGB, CMY, CMYK, YIQ, HSV (HSB) i HLS model.

****

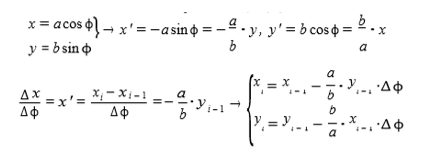
****

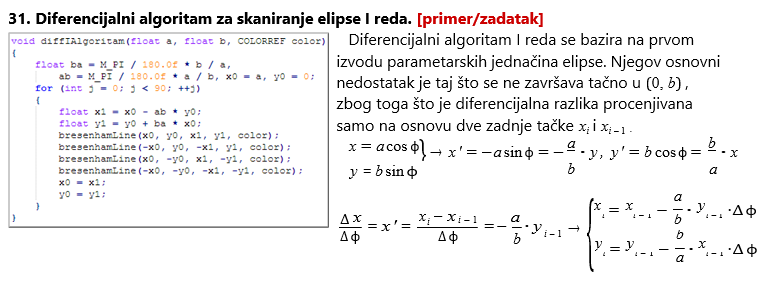
**8. Izvesti i dati realizaciju u c-u diferencijalnog algoritma ll reda za skaniranje elipse.**

**2. Izvesti i dati realizaciju u c-u diferencijalnog algoritma l reda za skaniranje elipse**.



Diferencijalni algoritam I reda se bazira na prvom izvodu parametarskih jednačina elipse. Njegov osnovni nedostatak je taj što se ne završava tačno u (0, *b*) , zbog toga što je diferencijalna razlika procenjivana samo na osnovu dve zadnje tačke *xi* i *xi* − 1 .

**

**

**6. Objasniti iscrtavanje ekrana sa i bez preplitanja**

**Bez preplitanja** (non-interlaced). Postoji samo jedan frejm i on se iscrtava u svakom frejm periodu. U frame baferu je frame koji se iscrtava kolko god puta je potrebnom (ceo frame se iscrta).

**Sa preplitanjem** (interlaced). Postoje dva frejma. U jednom su parne, a u drugom neparne linije. U jednoj poluperiodi frejma se iscrtava jedan frejm, a u drugoj poluperiodi se iscrtava drugi frejm. Ovaj metod daje dobre rezultate sa monitorima koji rade na nižim frekvencijama

**9. Razlika izmedju rasterskih i vektorskih podataka. Opisati način rada rasterskih i vektorskih uredjaja.**

Rasterski podaci se pamte u obliku matrice čiji je svaki element jedan piksel zapamćene slike, dok se vektorski podaci pamte kao skup tačaka i primitiva na osnovu kojih se dobija zapamćena slika.

Rasterski podaci predstavljaju „digitalni pogled“ na izvorni dokument, i iz njih se direktno mogu izdvajati informacije korišćenjem neke od metoda za obradu signala. Takođe, oni se mogu kompresovati, što smanjuje potreban prostor za njihovo skladištenje. Nedostatak je to što se mora izvršavati dekompresija, što usporava prikaz.

Prednost vektorskih podataka u odnosu na rasterske je u tome što pružaju mnogo više informacija o geometriji i topologiji, kao i to što prilikom uvećanja/umanjenja slika ne gubi kvalitet. Zato se vektorski podaci koriste za pamćenje crteža, dok se rasterski podaci koriste za pamćenje slike, videa itd., tamo gde njihova ograničena preciznost ne dolazi do izražaja.

# **Razlika između rasterskih i vektorskih uređaja.**

Displej procesor vektorskih uređaja interpretira komande za iscrtavanje iz bafera (često se zove bafer za osvežavanje - *refresh buffer*) i šalje digitalne koordinate tačke generisanjem vektora koji se konvertuje u odgovarajući analogni signal, koji predstavlja napon potreban da bi se osvetlile tačke na ekranu. Zrak se kreće u onom redosledu i smeru u kome mu diktiraju koordinate iz bafera. Tehnika prikaza na vektorskim uređajima se naziva slučajnim skaniranjem (*random scan*) jer je svaka tačka na ekranu adresibilna.

Rasterski uređaji smeštaju podatke za prikazivanje u bafer za osvežavanje, pri čemu su primitive razložene na piksele. Kompletna slika na rasterskom uređaju se formira iz rastera, koji predstavlja niz horizontalnih rasterskih linija od kojih je svaka red individualnih piksela. Raster se pamti kao matrica piksela koja predstavlja celu površinu ekrana. Slika se skenira sekvencijalno od strane video kontrolera, liniju po liniju i od vrha ka dnu. Intenzitet mlaza reflektuje intenzitet svakog piksela.

**13. Navesti modele senčenja koji se koriste u rac.gr. Detaljno objasniti Flat model. Šta su dobre/loše strane modela?**

Lokalni pristup:

Konstantno senčenje (Flat shading).

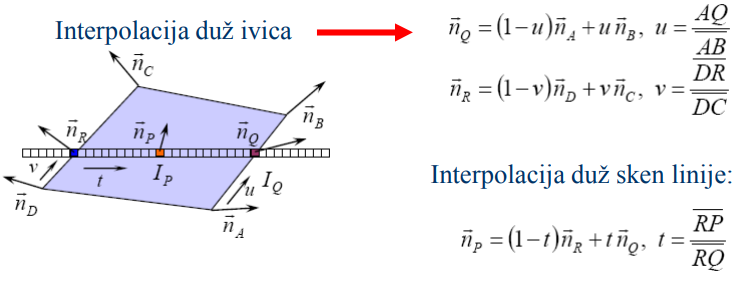
Gouraud-ovo senčenje.

Phong-ovo senčenje.

Flat: Sve tačke površine unutar jednog poligona imaju isti intenzitet. Dobra strana ove metode je velika brzina, ali je veliki nedostatak prevelika istaknutost ivica.

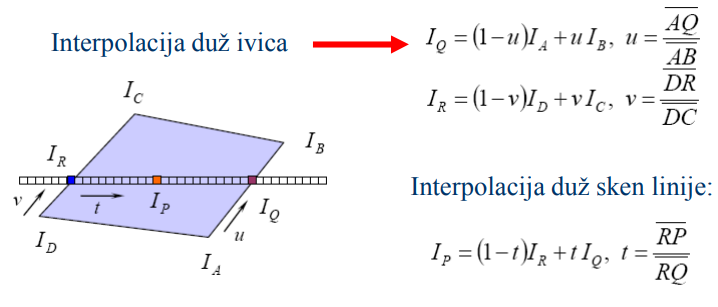
**17. Navesti modele senčenja koji se koriste u rac.gr. Detaljno objasniti Phong model. Šta su dobre/loše strane modela?**

Phong: Kod ovog načina se radi interpolacija normala, tako da se izračunavanje boje i intenziteta obavlja za svaki piksel ponaosob, što za rezultat ima veliki broj izračunavanja i malu mogućnost primene za senčenje u realnom vremenu .

****

**23. Navesti modele senčenja koji se koriste u rac.gr. Detaljno objasniti Gouraud model. Šta su dobre/loše strane modela?**

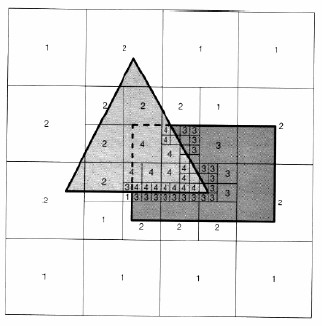
Gouraud: Ova metoda eliminiše vidljivost ivica interpolacijom boja i intenziteta osvetljenosti duž svakog poligona u 3D površi. Boja i intenzitet se računaju za svako teme poligona a zatim se vrši interpolacija duž ivica poligona. Interpolacija se vrši još jednom duž sken-linija prilikom skeniranja svakog od poligona. Rezultat ovakvog postupka su vrlo glatki prelazi. Postupak je sporiji od Flat, ali još uvek dovoljno brz za praktičnu upotrebu.

****

**27. Objasniti Warnock-ov i Watkinson-ov alg.**

*Warnock*-ov algoritam se može predstaviti u tri koraka:

1. Analizira se sadržaj prozora koji se ispituje (u početku je taj prozor jednak zaslonu), pri čemu se mogu javiti sledeći slučajevi:

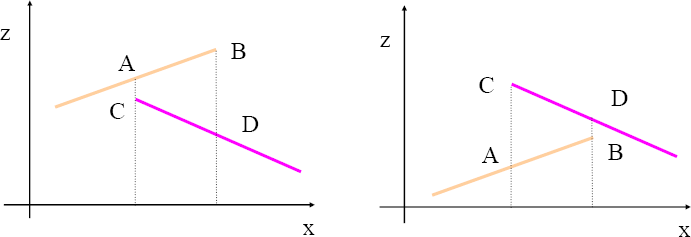
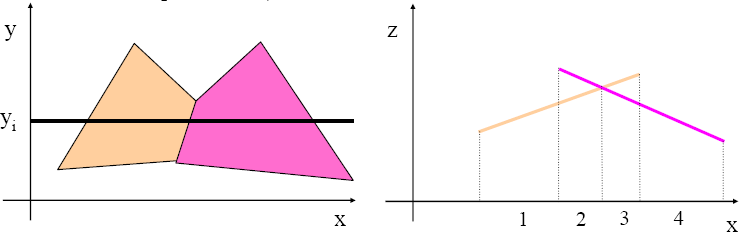


* + 1. Prozor je prazan.
    2. Scena u prozoru je prosta i moguće ju je prikazati.
    3. Scena u prozoru je složena, pa je potrebno rekurzivno podeliti prozor i primeniti algoritam.

1. Deli se prozor po *quadtree* principu.
2. Dele se poligoni u tri grupe, u zavisnosti od relacije sa prozorom:
   * 1. Poligon je izvan prozora, pa se uklanja iz liste.
     2. Poligon seče prozor ili je u njemu, pa se čuva delom ili u celosti.
     3. Poligon prekriva prozor, pa se ceo prozor boji bojom poligona.

*Scan-line(*Watkinson-ov*)* algoritamse može predstaviti u 4 koraka:

1. Postavlja se ispitna linija u projekcionoj ravni.
2. Određuju se tzv. „rasponi uzorka“ – delovi linije na kojima se ne može dogoditi promena vidljivosti. Oni zadovoljavaju sledeće uslove:
   * 1. Broj segmenata u rasponu uzorka je konstantan I veći od nule.



1. Projekcije preseka za *y* = *yi* unutar raspona uzorka u ravni *xz* se ne seku unutar raspona uzorka - svaki presek u projekciji označava noviraspon uzorkaOdređuje se vidljivost - raspon uzorka se testira u odnosu na vidljivost u dva slučaja:
   1. Ako je broj segmenata na tekućoj ispitnoj liniji različit od broja segmenata na prethodnoj ispitnoj liniji.
   2. Ako se krajnje tačke dva segmenta zamene, po veličini *z* koordinate (npr. tačke *A* i *C* , *B* i *D* ) kad prelazimo iz tekuće u susednu ispitnu liniju - tada se obe površine seku u prostoru između dve ispitne ravni.
2. Ispitivanjem *z* koordinate možemo odrediti koji segment u rasponu uzorka prekriva druge segmente.

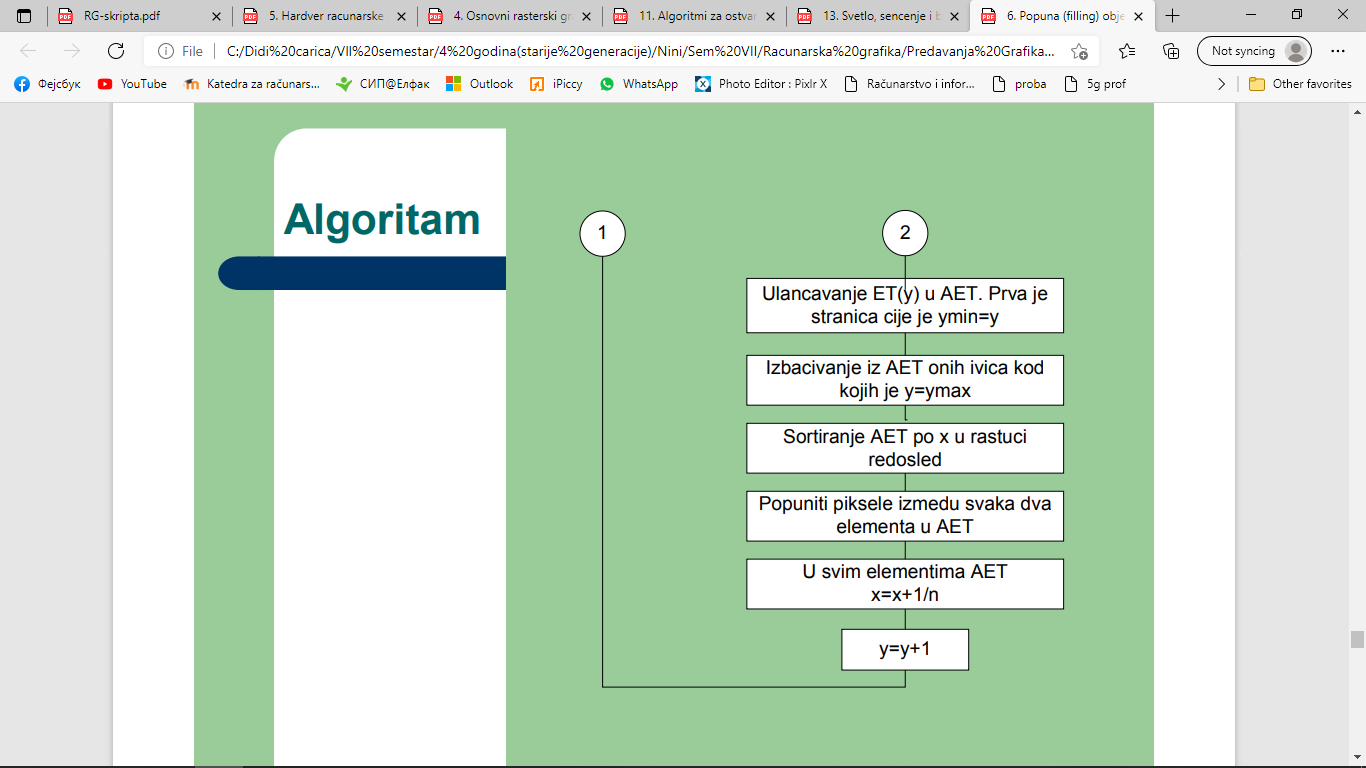
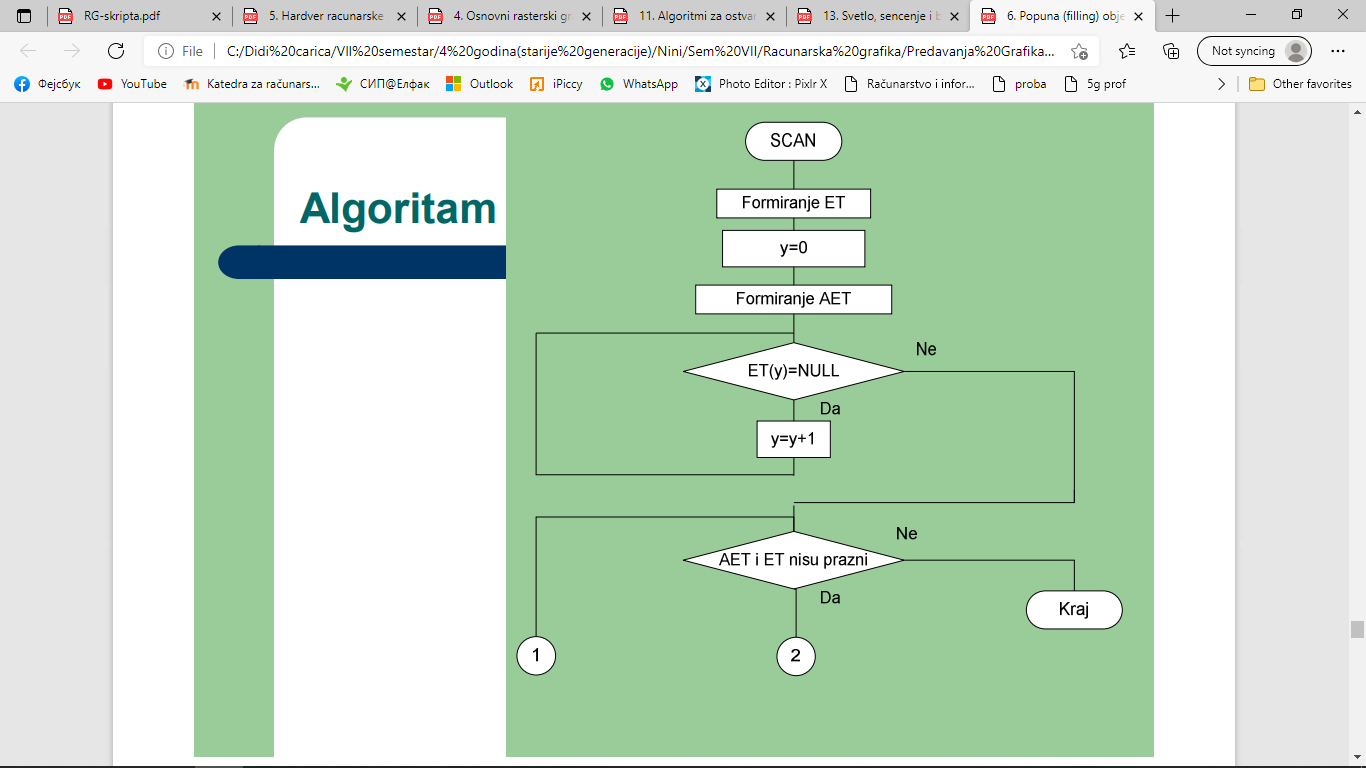
**11. Navesti algoritme za uklanjanje sakrivenih ivica i površi. Objasniti Z-buffer alg. Navesti dobre i lose strane algoritma.**

Algoritmi za uklanjanje sakrivenih ivica i površi dele se na sledeće 2 grupe:

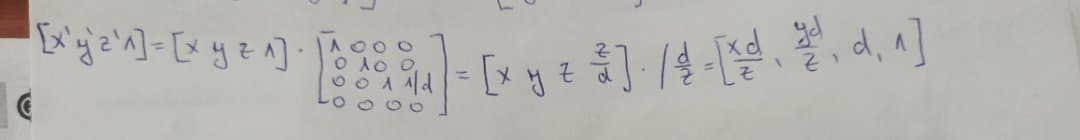
1. Algoritmi prostora slike (*image-precision*): Za svaki piksel slike dimenzija *m* × *n* određuju koji je od *k* objekata vidljiv - složenost je *O* (*m* ⋅ *n*⋅ *k*) i u njih spadaju:
   * 1. *Z-buffer* algoritam.
     2. *Warnock*-ov algoritam.
     3. *Scan-line* (*Watkinson*-ov) algoritam.
     4. *Ray-tracing* algoritam.
2. Algoritmi prostora objekata (*object-precision*): Za svaki od *k* objekata u sceni određuje koji je vidljiv - složenost je *O* (*k*2) i u njih spadaju:
   * 1. Slikarev algoritam.
     2. *Back face culling* algoritam.
     3. Algoritam koji koristi BSP stabla.
     4. Algoritam koji koristi *octree* stabla.

Algoritam sa Z baferom koristi pored bafera uređaja za prikazivanje slike (frame buffer) i poseban bafer gde se u poziciji (x,y) pamti najmanja Z koordinata (dubina) tačaka koje se vide iz piksela (x,y). Ovaj bafer se zove Z-buffer. Dakle, najmanja dubina svih tačaka koje se vide iz piksela (x,y) se pamti u Z baferu na poziciji (x,y). Koristi se u OpenGL-u

**25. Napisati algoritam za popunu poligona zadatog listom temena.**

****

**20. Izvesti matricu za perspektivnu projekciju ukoliko se centar projekcije nalazi u koordinatnom početku, a projekciona ravan je ravan z=2.**

****

**18. Navesti sve logičke tipove ulaznih i izlaznih uredjaja. Objasniti namenu svakog tipa uredjaja.**

Prema njihovoj ulozi u sistemu, svi ulazni uređaji se mogu svrstati u:

– Lokator (pokazuje poziciju i orijentaciju) - grafička tabla, miš, džojstik, Touchpad , Trackball, Ekrani osetljivi na dodir

– Valuator (za unos jednog realnog broja) – potenciometar.

– Tastatura (za unos znakovnog niza).

– Pokazivač (za izbor elemenata slike) - svetlosno pero (stilus).

– Dugme (za izbor neke akcije iz skupa mogućih alternativa).

– Uređaji za unos slike, videa i 3D modela: U ovu grupu spadaju:

\* Za unos slika: dvodimenzionalni (2D) skener, digitalni fotoaparati.

\* Za unos video podataka: digitalne i web kamere.

\* Za unos 3D modela: trodimenzionalni (3D) skener

– Uređaji za unos zvuka: ubrajamo: mikrofone, digitalne diktafone i digitalne snimače zvuka

– VR ulazni uređaji : Poslednju grupu ulaznih uređaja čine uređaji koji se koriste u oblasti virtuelne realnosti (VR). Tu spadaju različite vrste VR rukavica kao i različite vrste uređaja za detekciju ljudskih pokreta.

Izlazni uređaji služe za praćenje rada računara kao i za predstavljanje rezultata obrade u formi razumljivoj za korisnika. Generalno, sve izlazne uređaje možemo podeliti na :

– ekrane (monitore, displeje),

– projektore,

– štampače,

– zvučnike i

– VR izlazne uređaje

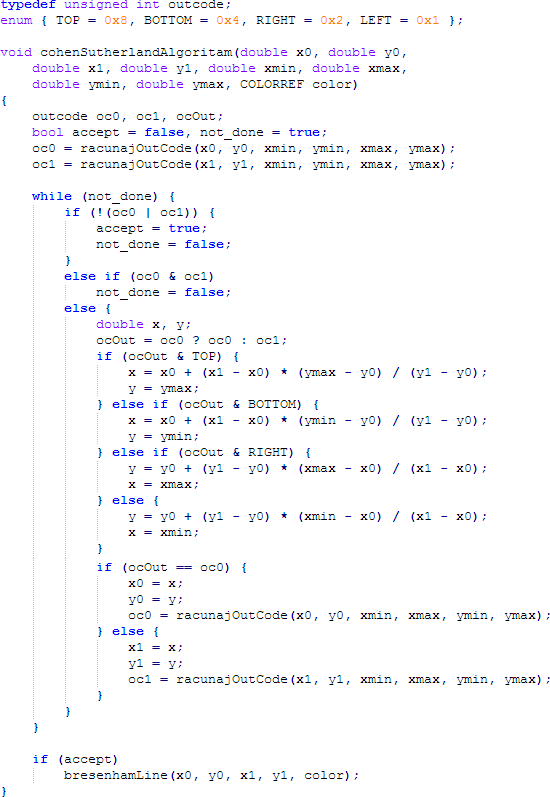
**24. Definisati grafički API. Nabrojanih i objasniti osnovne grupe funkcija koje bi svaki dobar grafički API trebalo da ima.**

Grafički API (*Application Programming Interface*) je skup grafičkih funkcija organizovanih u jednu ili više grafičkih biblioteka koje predstavljaju interfejs između aplikacionog programa i grafičkog sistema.

Grafički API vrši apstrakciju detalja o hardverskoj i softverskoj implementaciji i daje programeru prikaz sistema kao „crne kutije“.

1. Funkcije za primitive: funkcije za iscrtavanje osnovnih grafičkih primitiva (tačka, linija, poligon, piksel, tekst…).
2. Funkcije za kontrolu atributa primitiva: Primitive definišu šta će se nacrtati a njihovi atributi kako će se nacrtati (npr. boja/debljina/tip linije).
3. Funkcije pogleda: Omogućavaju formiranje 2D prikaza na osnovu 3D objekta (projekciju, određivanje koji objekti iz 3D sveta će se naći u 2D prikazu).
4. Funkcije za geometrijske transformacije: Omogućavaju 2D i 3D geometrijske transformacije nad objektima (translacija, rotacija, skaliranje...).
5. Funkcije za ulaz: Omogućavaju korisnicima da vrše unos podataka u aplikaciju putem delova grafičkog sistema (tastatura, miš, tabla, skener…).
6. Kontrolne funkcije: Omogućavaju komunikaciju sa operativnim sistemom pod kojim se grafička aplikacija izvršava (inicijalizacija, kontrola prozora...).
7. Ispitivačke funkcije: Omogućavaju dobijanje informacija o karakteristikama pojedinih komponenata grafičkog sistema (paleta boja, rezolucija…). Na ovaj način programeri dobijaju mogućnost da pišu device independent aplikacije koje ne zavise od grafičkog sistema na kome se izvršavaju.

**19. Objasniti Cohen-Sutherlandov alg za odsecanje linija i dati realizuju u C-u.**



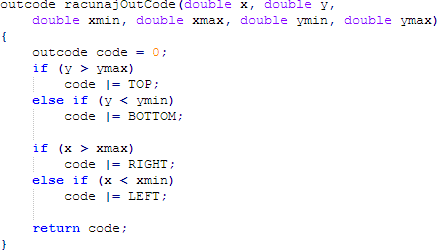
Osnovna ideja *Cohen-Sutherland* algoritma je da se najpre pokuša prihvatanje ili odbacivanje linije u celini. Ako to ne urodi plodom, određuje se presek linije i produžene ivice prozora pa se ponovo pokušava prihvatanje ili odbacivanje ostatka linije. Posmatraju se krajnje tačke linija, pri čemu se razlikuju tri slučaja:

1. Ispunjen je uslov trivijalnog prihvatanja: Obe krajnje tačke su unutar regiona za odsecanje.
2. Ispunjen je uslov trivijalnog odbacivanja: Obe krajnje tačke su levo od *xl* ili desno od *xr* , ili iznad *yt* ili iznad *yb* .
3. Svi ostali slučajevi: Neophodno je izračunavanje preseka.

Prilikom izračunavanja se ivice prozora produžavaju, tako da se cela slika sastoji iz 9 oblasti. Svakoj oblasti se pridružuje 4-bitni položajni kod (*outcode*) *b*3 *b*2 *b*1 *b*0 , pri čemu svaki bit označava jednu oblast ( *b*3 - iznad, *b*2 - ispod, *b*1 - desno i *b*0 - levo). Zatim se određuju položajni kodovi krajnjih tačaka linije. Uslov trivijalnog prihvatanja ispunjen je ako su oba koda

imaju bar jedan zajednički bit. U svim drugim slučajevima se za krajnju tačku linije koja nije u prozoru ispituje da li je levo, desno, iznad ili ispod prozora, te se nalazi presek linije koja se odseca sa odgovarajućom produženom ivicom prozora.

Tada se krajnja tačka linije premešta u tačku preseka, i izračunava joj se položajni kod, nakon čega se ponavlja ispitivanje uslova.



**29. Objasniti koncept grafičkog uredjaja (Device Context)**

**-Device Context.**

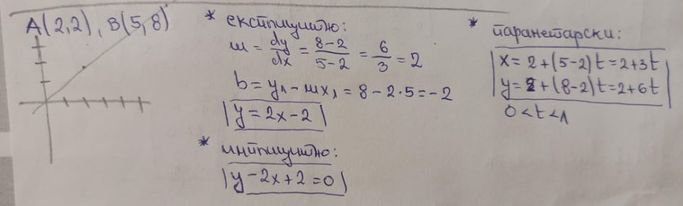
*Device Context* je apstraktni (virtuelni) uređaj. Kada se programira, pristupa se ovom uređaju i na taj način se ne vodi računa direktno o hardveru i njegovim karakteristikama. O karakteristikama hardvera brigu vode drajveri uređaja.

**12. Pravu koja je odredjena tačkama A(2,2) i B(5,8) predstaviti u:**

**a. Eksplicitnom**

**b. Implicitnom**

**c. Parametarskom obliku**

****

**26. Bresenhamov alg za liniju- Naći piksele koji će biti selektovani prilikom crtanja linije (4,4), (8,16).**

**21. Izračunati matricu za 2D refleksiju u odnosu na pravu x\*y=0.**

**30. Izračunati matricu za 2D refleksiju u odnosu na pravu y=-x.**

**15. Izračunati matricu za 2D refleksiju u odnosu na pravu 2\*y - 2\*x=0**

**10. Izračunati matricu za 2D refleksiju u odnosu na pravu x+y=0.**

**28. Izvesti transformacionu matricu za rotaciju oko tačke (Dx, Dy) za ugao od -90°.**

**16. Pravu koja je odredjena tačkama A(4,4) i B(10,16) predstaviti u:**

**a. Eksplicitnom**

**b. Implicitnom**

**c. Parametarskom obliku**

**22. Slika BSP stablo (februar 2018)**