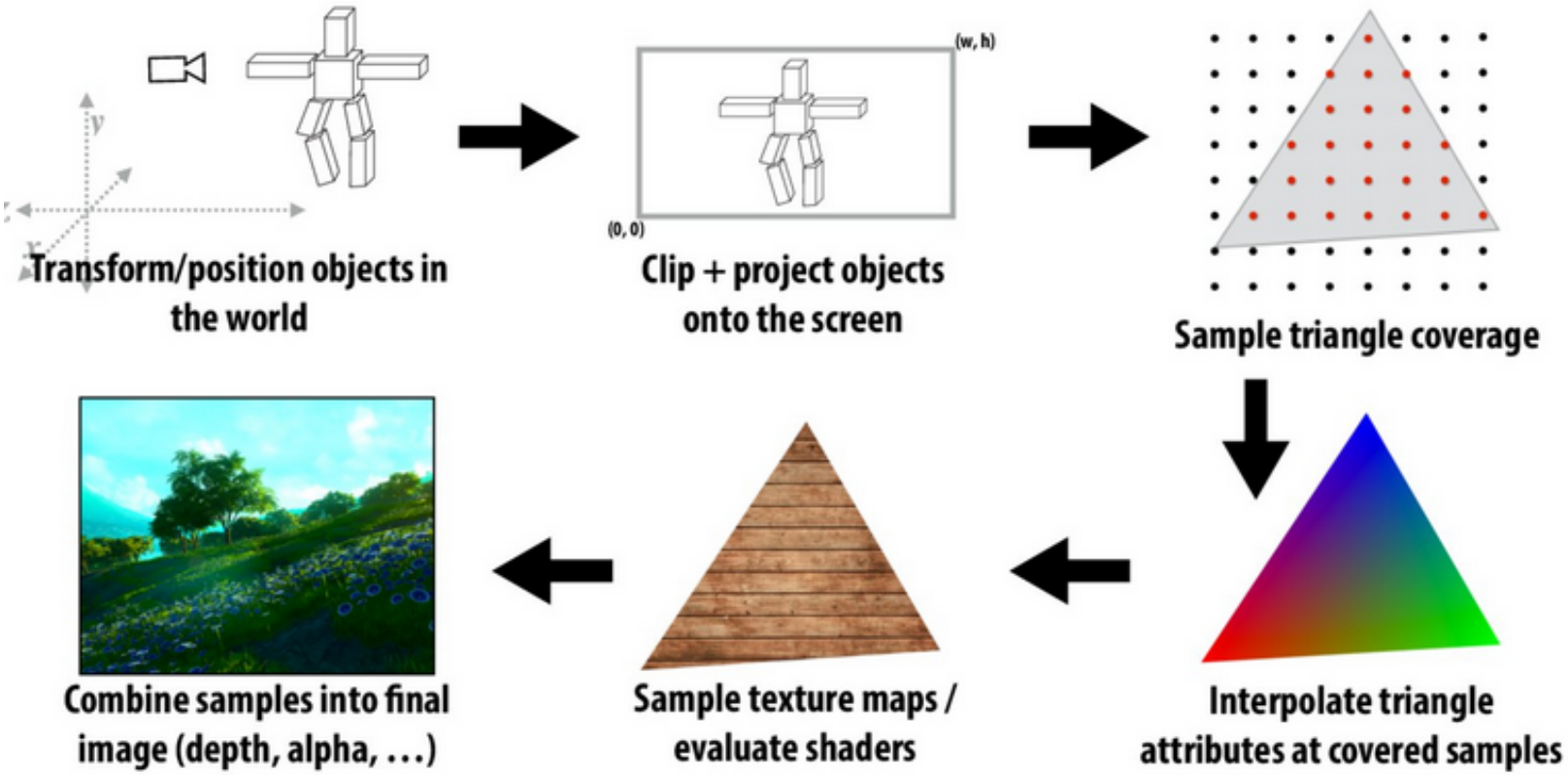
Računarska grafika
Vidljivost

Vesna Marinković

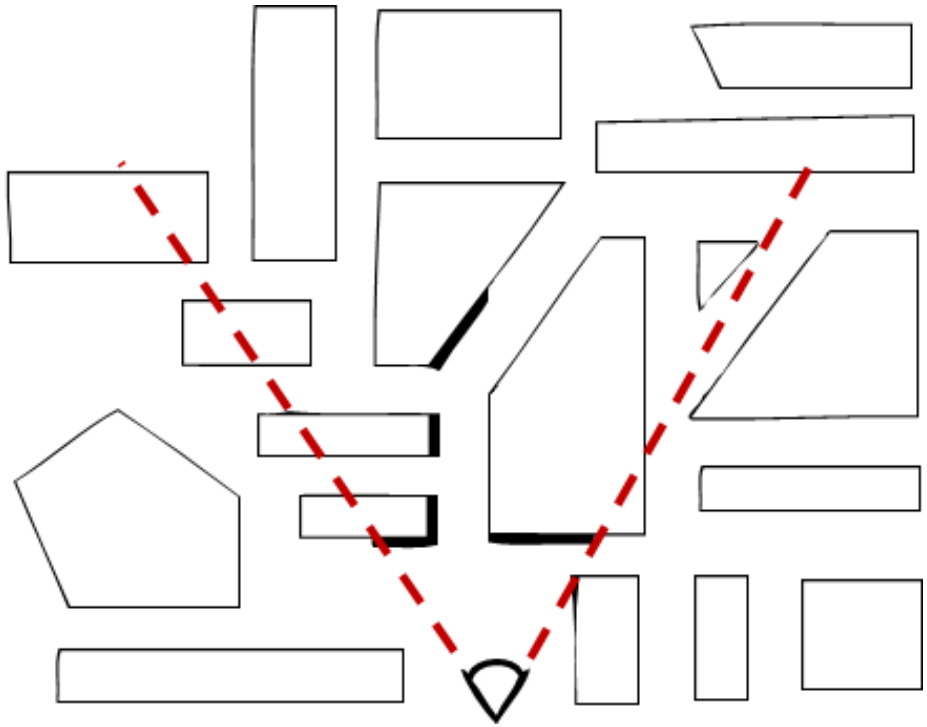
Podsetimo se: protočna obrada kod rasterizacije



Fokus ovog predavanja je na poslednjoj etapi obrade

Problem utvrđivanja vidljivosti

- Renderovati ili ne, pitanje je sad...

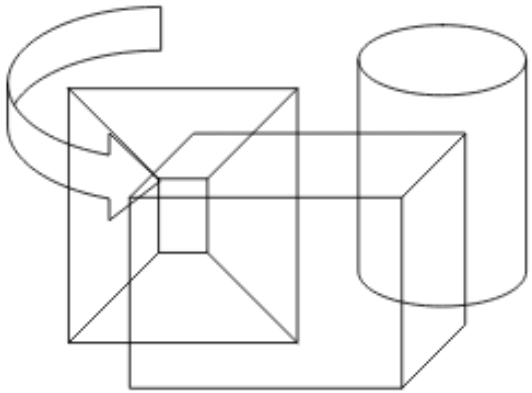


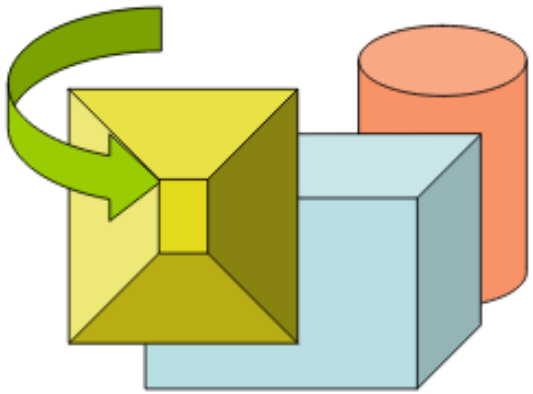
Problem utvrđivanja vidljivosti

- Problem utvrđivanja vidljivosti se bavi utvrđivanjem za dati piksel koja je površ na sceni vidljiva
- Fundamentalni problem u računarskoj grafici
- Poznat je i pod nazivima
  - određivanje vidljivih površi (visible surface derermination (VSD))
  - odbacivanje skrivenih površi (hidden surface removal (HSR))
- OpenGL se bavi ovim problemom razmatrajući fragment po fragment
- Međutim, postoje metode za odbacivanje skrivenih površi koje se izvršavaju pre nego što se procesom rasterizacije poligon konvertuje u fragmente

Klasifikacija algoritama prema Saderlandu, 1973.

- Algoritmi prostora slika (image-precision algoritmi)
  - rade nad projekcijama objekata u koordinatama ekrana
  - za svaki piksel određuje se koji od objekata je vidljiv, tj. najbliži posmatraču i piksel se boji odgovarajućom bojom
- Algoritmi prostora objekta (object-precision algoritmi)
  - rade direktno nad objektima u svetskom koordinatnom sistemu
  - za svaki objekat određuje se deo objekta koji je vidljiv
  - objekti se međusobno porede i eliminišu se kompletni objekti ili delovi objekta koji nisu vidljivi





Vremenska složenost različitih klasa algoritama

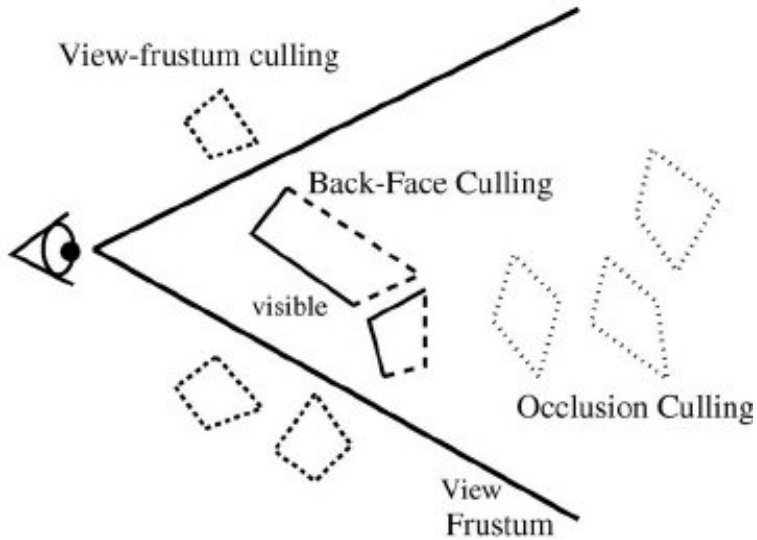
- Algoritmi prostora slika
  - složenost = $O($ broj piksela $\cdot$ broj objekata$)$
  - npr. za $1286 \cdot 1024$ piksela i 1 milion objekata je $O(10^{12})$
- Algoritmi prostora objekta
  - rezolucija uređaja za prikaz nije relevantna jer se izračunavanja izvode na nivou objekata
  - u najgorem slučaju potrebno je porediti svaki objekat sa svakim
  - složenost = $O(n^2)$, n - broj objekata
  - npr. za 1 milion objekata je $O(10^{12})$

Klasifikacija algoritama za utvrđivanje vidljivosti

- Algoritmi za određivanje vidljivosti se na osnovu odnosa kvaliteta rezultata i brzine mogu klasifikovati na:
  - tačne (precizne) – garantuju dobijanje tačnog rezultata
  - konzervativne – odbacuju se delovi scene koji sigurno nisu vidljivi
- Često se vrši kombinovanje oba pristupa

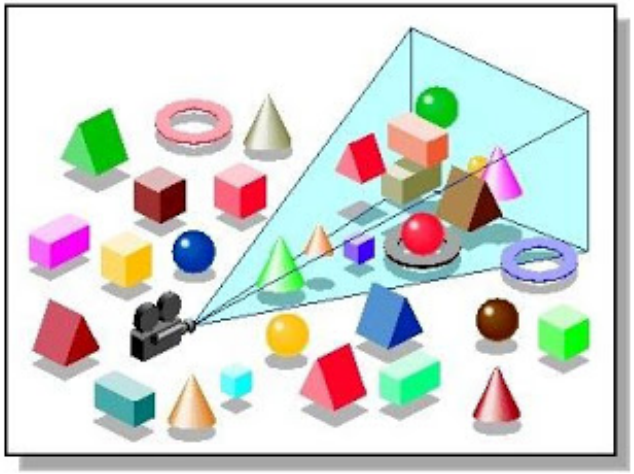
Klasifikacija algoritama za utvrđivanje vidljivosti

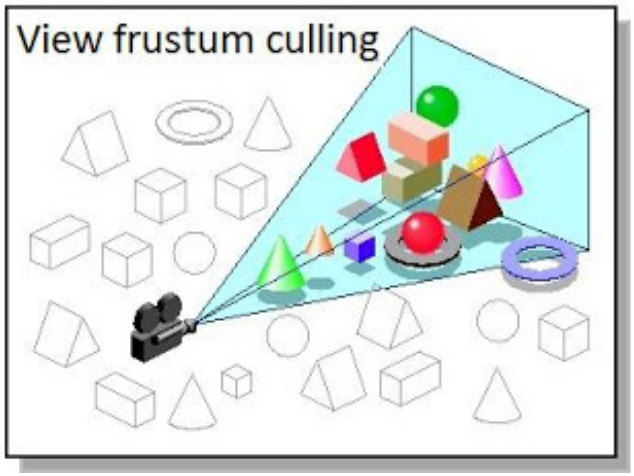
- Algoritmi koji garantuju tačnost
  - z-bafer algoritam
  - rej kasting algoritam
- Konzervativni algoritmi
  - odbacivanje dela van zapremine pogleda (view-frustum culling)
  - odbacivanje zadnje strane (naličja) objekta (back-face culling)
  - blokirajuće odbacivanje (occlusion culling)



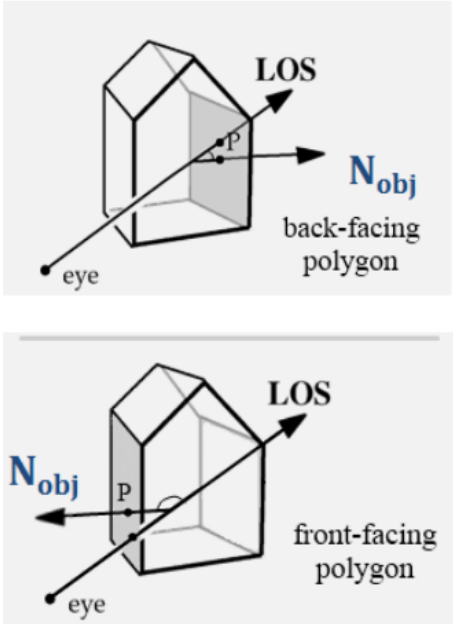
Odbacivanje dela van zapremine pogleda

- Zapremina pogleda je zadata jednačinama šest ravni, potrebno je proveriti da li se poligon nalazi sa druge strane neke od ovih ravni
- Veoma korisno u slučaju kada scena sadrži veliki broj složenih modela od kojih je mali broj njih unutar zapremine pogleda
- Unapređenje: oko svakog modela opisujemo sferu ili kocku i proveravamo da li je ona vidljiva – omogućava se trivijalno odbacivanje celog objekta koji nije vidljiv



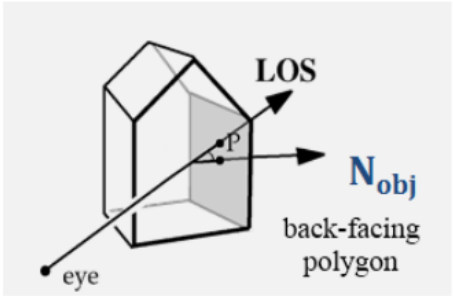


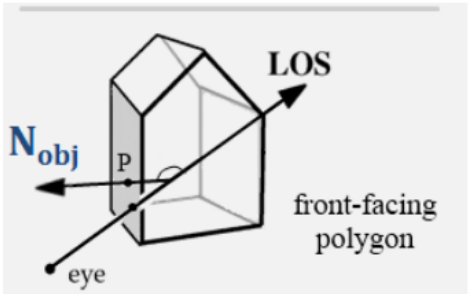
Odbacivanje zadnje strane objekta



- Zadnja strana neprozirnog čvrstog objekta nije vidljiva posmatraču
- Gotovo polovina geometrije scene se može odbaciti
- Pretpostavimo da je posmatrač van poliedra definisanog zatvorenom mrežom poligona i da gleda u smeru vektora $\overrightarrow{LOS}$
- Neka je P proizvoljna tačka poligona, a $\overrightarrow{N_{obj}}$ vektor normale poligona
- Kako odrediti zadnju stranu objekta?

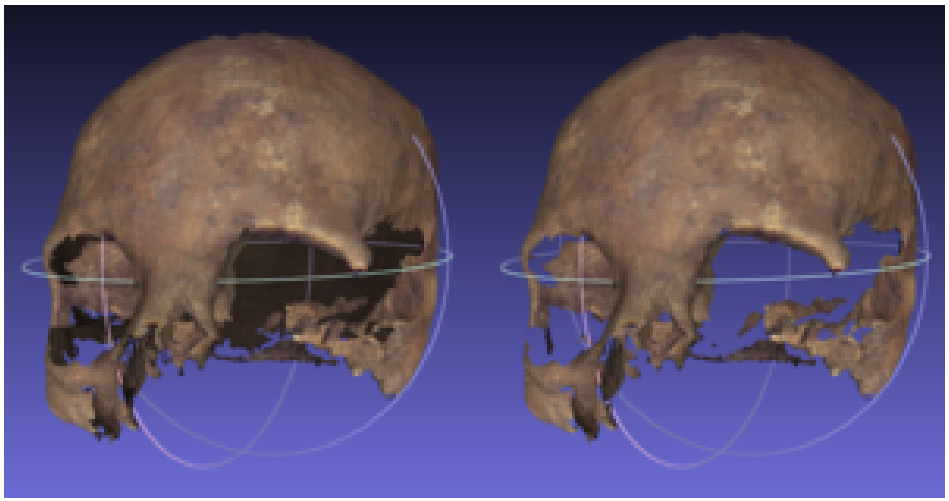
Odbacivanje zadnje strane objekta





- Poligon je:
  - sa prednje strane u odnosu na posmatrača ako je $\overrightarrow{LOS} \cdot \overrightarrow{N_{obj}} < 0$
  - sa zadnje strane u odnosu na posmatrača ako je $\overrightarrow{LOS} \cdot \overrightarrow{N_{obj}} > 0$
  - na konturi koja razdvaja prednju i zadnju stranu ako je $\overrightarrow{LOS} \cdot \overrightarrow{N_{obj}} = 0$

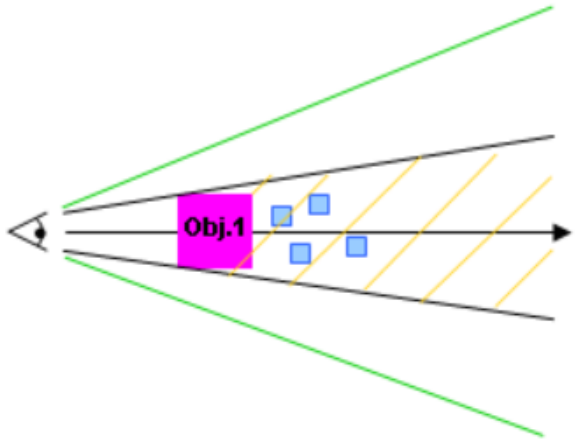
Odbacivanje zadnje strane objekta



- Trougao sa zadnje strane u odnosu na posmatrača nije vidljiv samo ako se nešto nalazi ispred njega
- Za ovo postoje garancije samo u slučaju zatvorenih, čvrstih tela
- Kod objekata koji sadrže “rupe” mogu se videti i poligoni sa zadnje strane u odnosu na posmatrača

Blokirajuće odbacivanje

- Cilj je efikasno identifikovati objekte zaklonjene drugim objektima
- Kombinuju se informacije o poziciji posmatrača sa relativnim pozicijama objekata da bi se izbeglo renderovanje kompletnih objekata ili hijerarhija scene
- Najčešće se koriste namenske prostorne strukture podataka

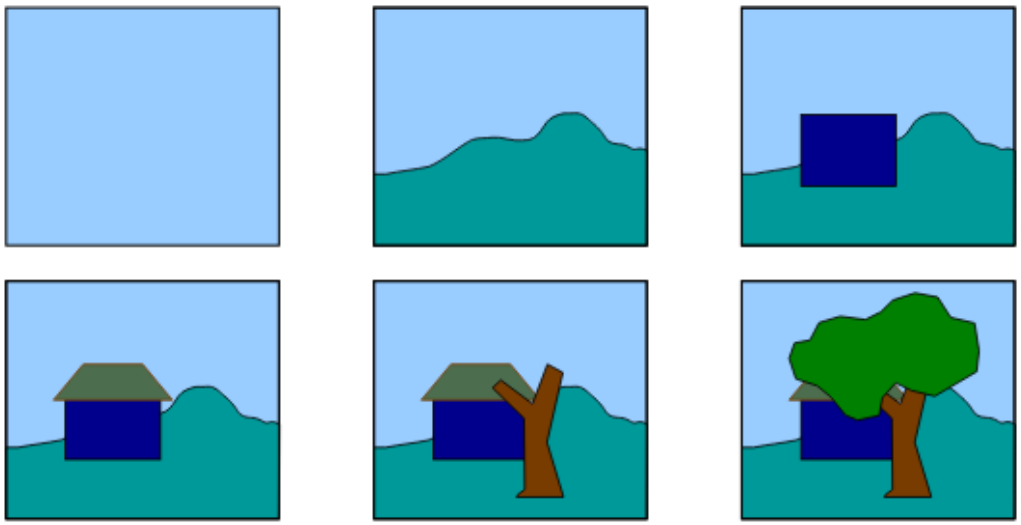


User's field of view

Area where objects hidden by **Object 1** are not displayed

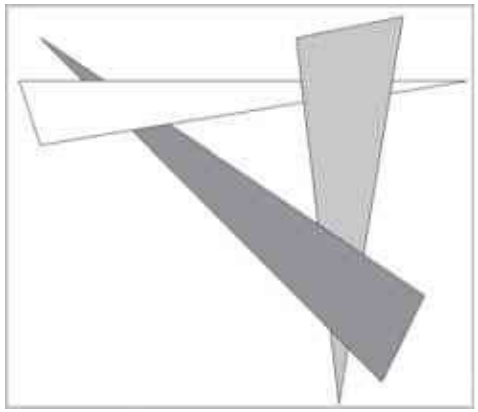
Algoritmi sa listama prioriteta

- Algoritmi sa listama prioriteta implicitno rešavaju problem vidljivosti tako što renderuju elemente scene u redosledu od udaljenijih ka bližim
- Dalji objekti imaju viši prioritet i kao takvi se prvi renderuju, a biće naknadno zaklonjeni objektima koji se kasnije renderuju
- Korišćeni su u prvom simulatoru letenja koji je računarski generisao slike u realnom vremenu (General Electric, 1967.)
- Danas se ovi algoritmi ne koriste često jer su razvijene bolje alternative
- Primer: slikarev algoritam



Slikarev algoritam

- Zaklanjanje i vidljivost se postižu tako što se boje daljih tačaka "pregaze" bojama bližih tačaka
- Ovu ideju možemo primeniti na:
  - pojedinačne piksele – neefikasno, ali tačno
  - kompletne primitive, npr. trouglove – efikasnije, ali postoje situacije kada ovo nije moguće uraditi
- Može se dozvoliti podela primitiva tamo gde se njihove projekcije seku



Algoritam sortiranja dubine

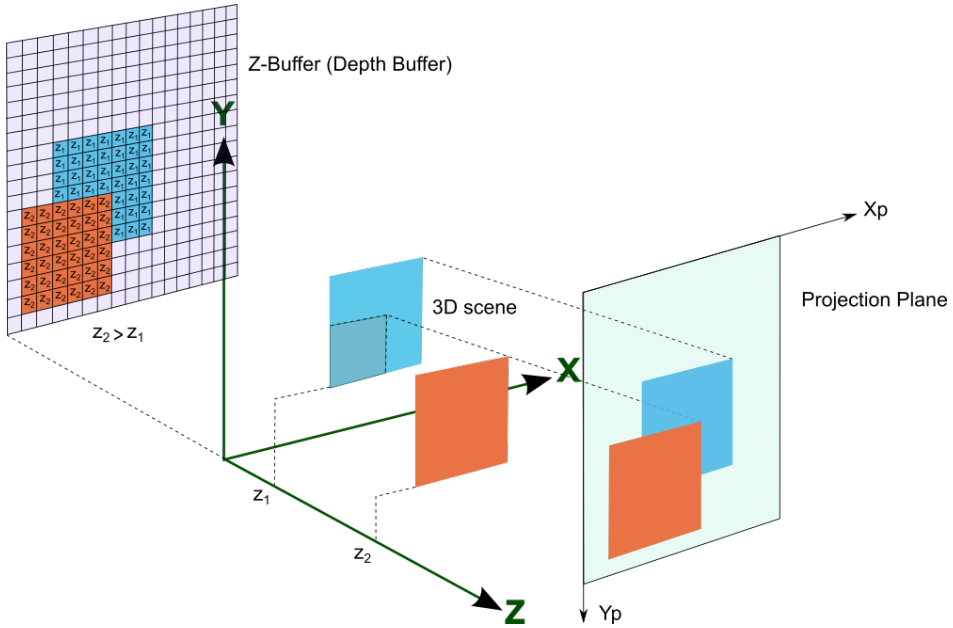
- Algoritam sortiranja dubine predstavlja modifikaciju slikarevog algoritma koja garantuje ispravno renderovanje
- Algoritam se sastoji od četiri koraka:
  1. svakom poligonu treba dodeliti ključ sortiranja koji je jednak z-vrednosti temena koje je najdalje od oblasti za prikaz
  2. sortirati poligone od najudaljenijeg do najbližeg prema vrednosti ključa
  3. otkriti slučajeve kada dva poligona imaju dvosmisleno uređenje – ovakve poligone deliti sve dok delovi nemaju eksplicitno uređenje i postaviti ih na pravo mesto u sortiranoj listi
  4. renderovati poligone u redosledu prioriteta, od najudaljenijeg do najbližeg

Bafer dubine

- Protočna obrada kod rasterizacije obrađuje trougao po trougao
- Na koji način uporediti više trouglova i saznati koji treba da se vidi u svakoj tački uzorkovanja?

Bafer dubine

- Protočna obrada kod rasterizacije obrađuje trougao po trougao
- Na koji način uporediti više trouglova i saznati koji treba da se vidi u svakoj tački uzorkovanja?
- Rešenje je održavanje bafera dubine
- Bafer dubine (ili z-bafer) za svaki piksel sadrži po jedan skalar koji daje neku meru rastojanja centra projekcije od površi koja boji piksel

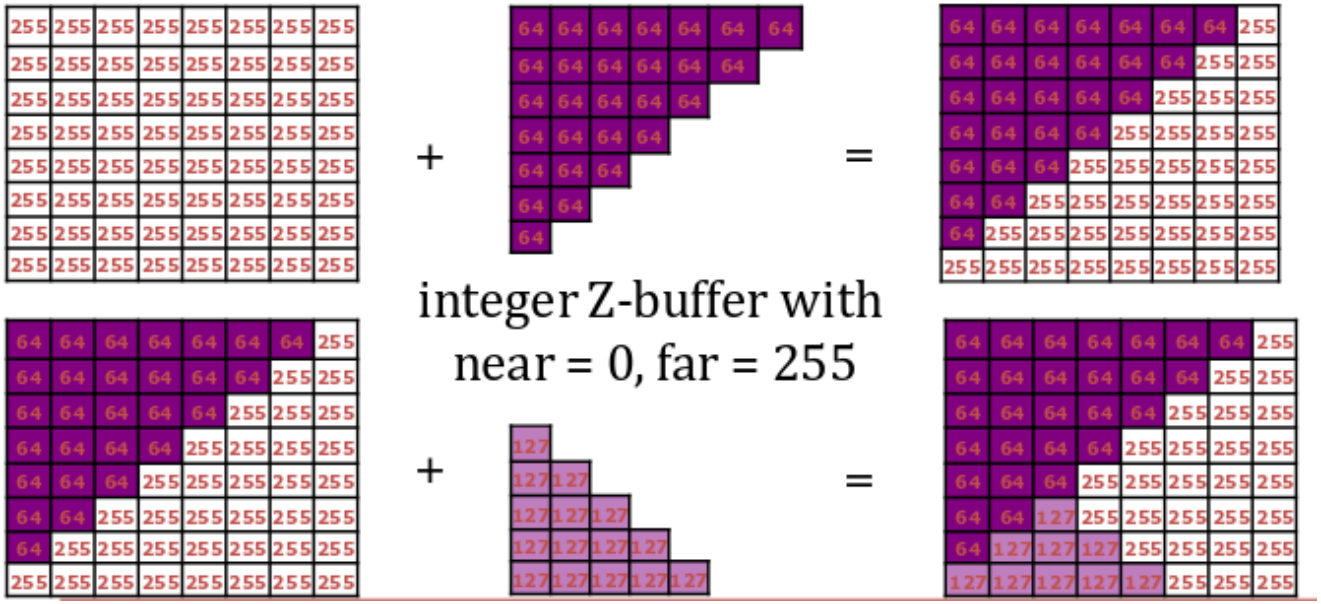


z-bafer algoritam

- Katmul, 1974.
- Sve vrednosti u kolor baferu se incijalizuju na boju pozadine, a sve vrednosti u z-baferu na z-koordinatu zadnje ravni odsecanja
- Maksimalna vrednost z-bafera je z-koordinata prednje ravni odsecanja
- Nekada se vrednosti u z-baferu čuvaju i kao celobrojne vrednosti

z-bafer algoritam

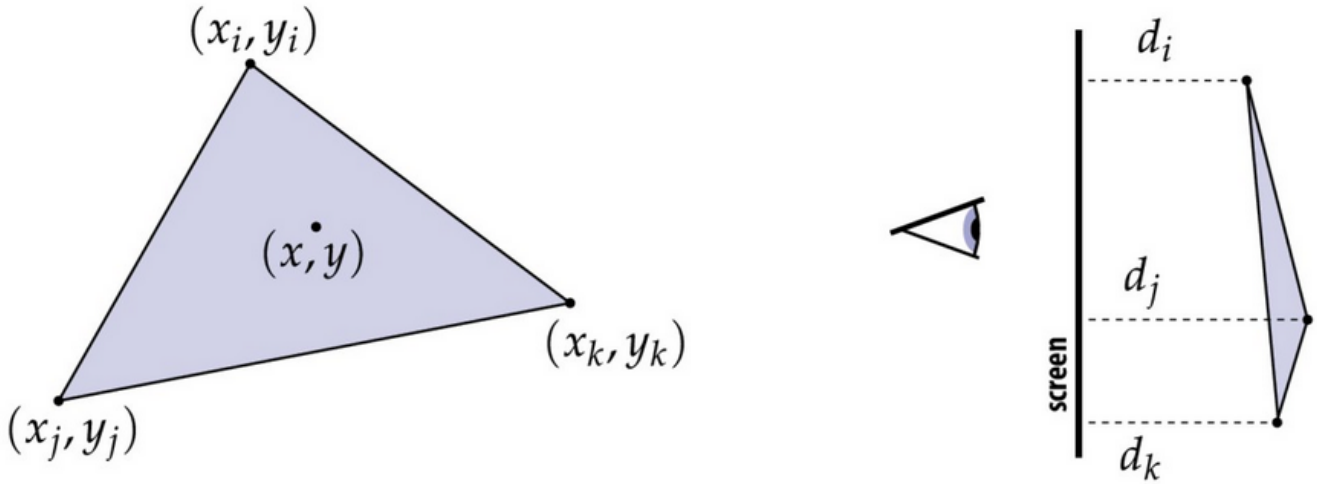
- Na sve trouglove koji predstavljaju scenu primenjuje se algoritam rasterizacije, u proizvoljnom poretku
- Prilikom rasterizacije za tačku (x, y), ako ta tačka nije dalja od posmatrača od tačke koja je trenutno u baferu, onda z koordinata te tačke zamenjuje postojeću vrednost u z-baferu, a njena boja se smešta u kolor bafer na poziciji (x, y)



Pseudokod algoritma

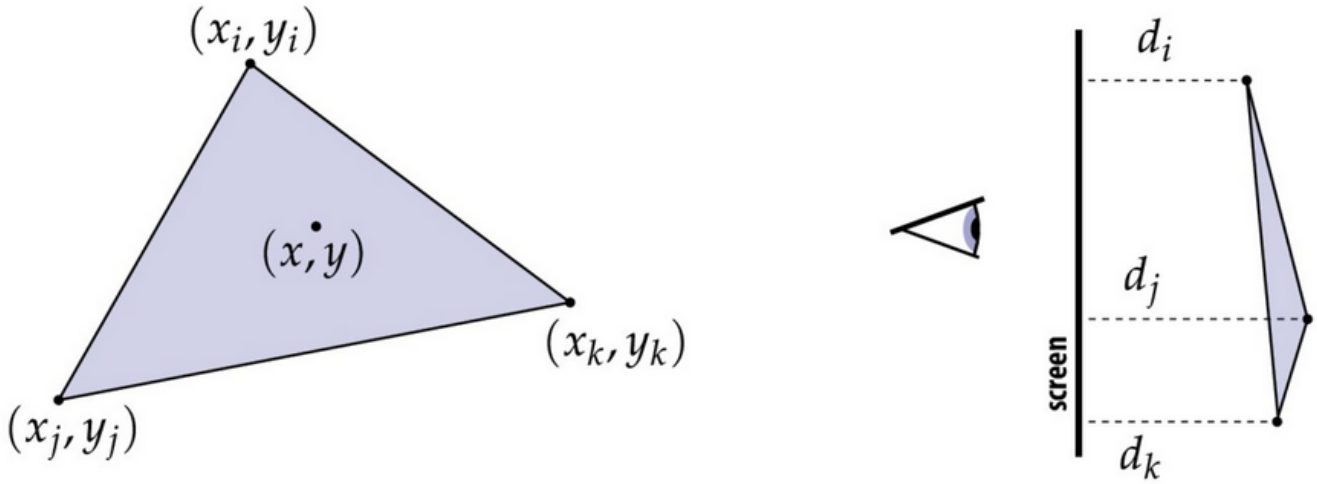
```
Procedure zBuffer
var
   pz: integer;
begin
 for y := 0 to YMAX do
  for x := 0 to XMAX do
   begin
    WritePixel(x,y,background_value);
    WriteZ(x,y,1)
   end

 za svaki poligon uradi sledece
  za svaki fragment 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.
```

Uzorkovanje dubine



- Pretpostavimo da je trougao zadat:
  - projektovanim 2D koordinatama (x_i, y_i) temena
  - dubinom d_i svakog temena (rastojanjem do posmatrača)
- Kako izračunati dubinu d u proizvoljnoj tački trougla (x, y) u kojoj vršimo uzorkovanje?

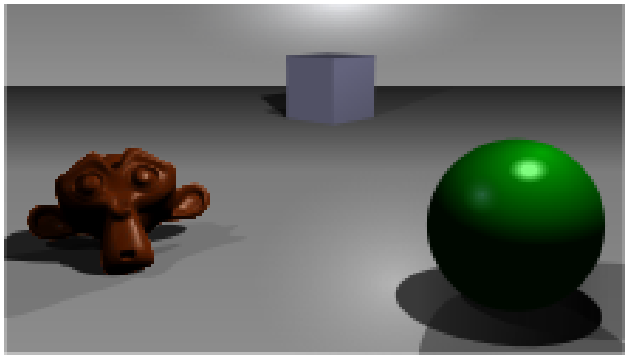
Uzorkovanje dubine



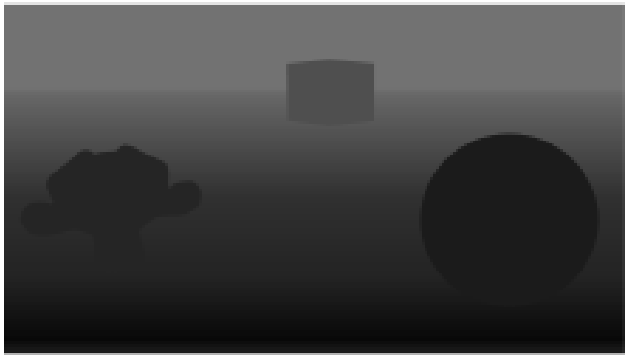
- Dubina se linearno menja duž trougla
- Vršimo interpolaciju vrednosti dubine u temenima korišćenjem baricentričnih koordinata (detaljnije naredne nedelje)

Ilustracija sadržaja z-bafera

- Za ilustraciju vrednosti u z-baferu koristimo nijanse sive
- Vrednosti bliske beloj su veoma daleko, a vrednosti bliske crnoj su blizu

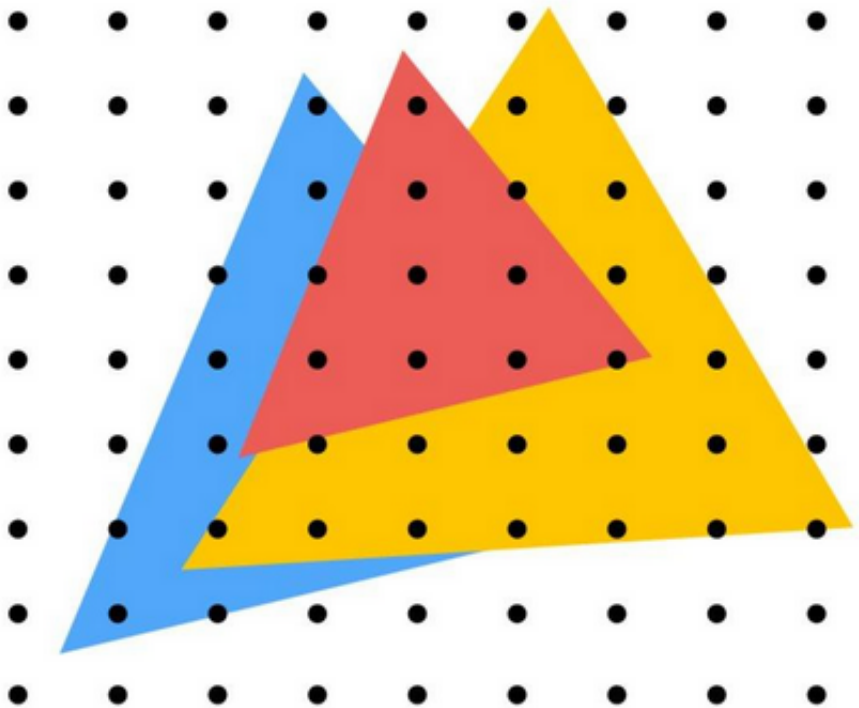


A simple three-dimensional scene



Z-buffer representation

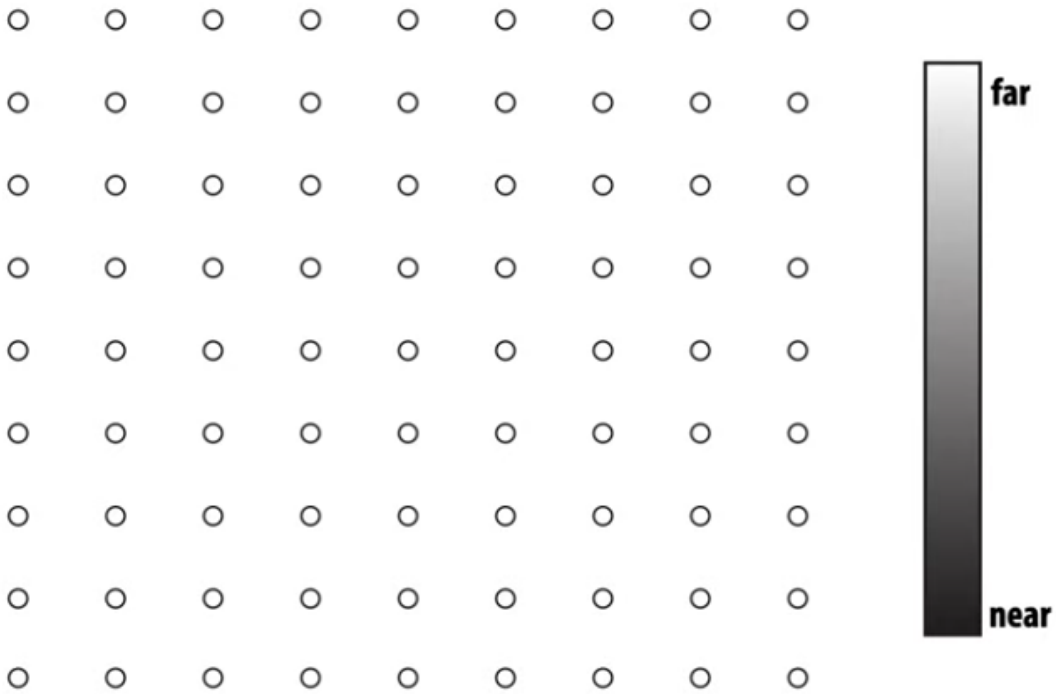
Primer: renderovanje tri neprozirna trougla



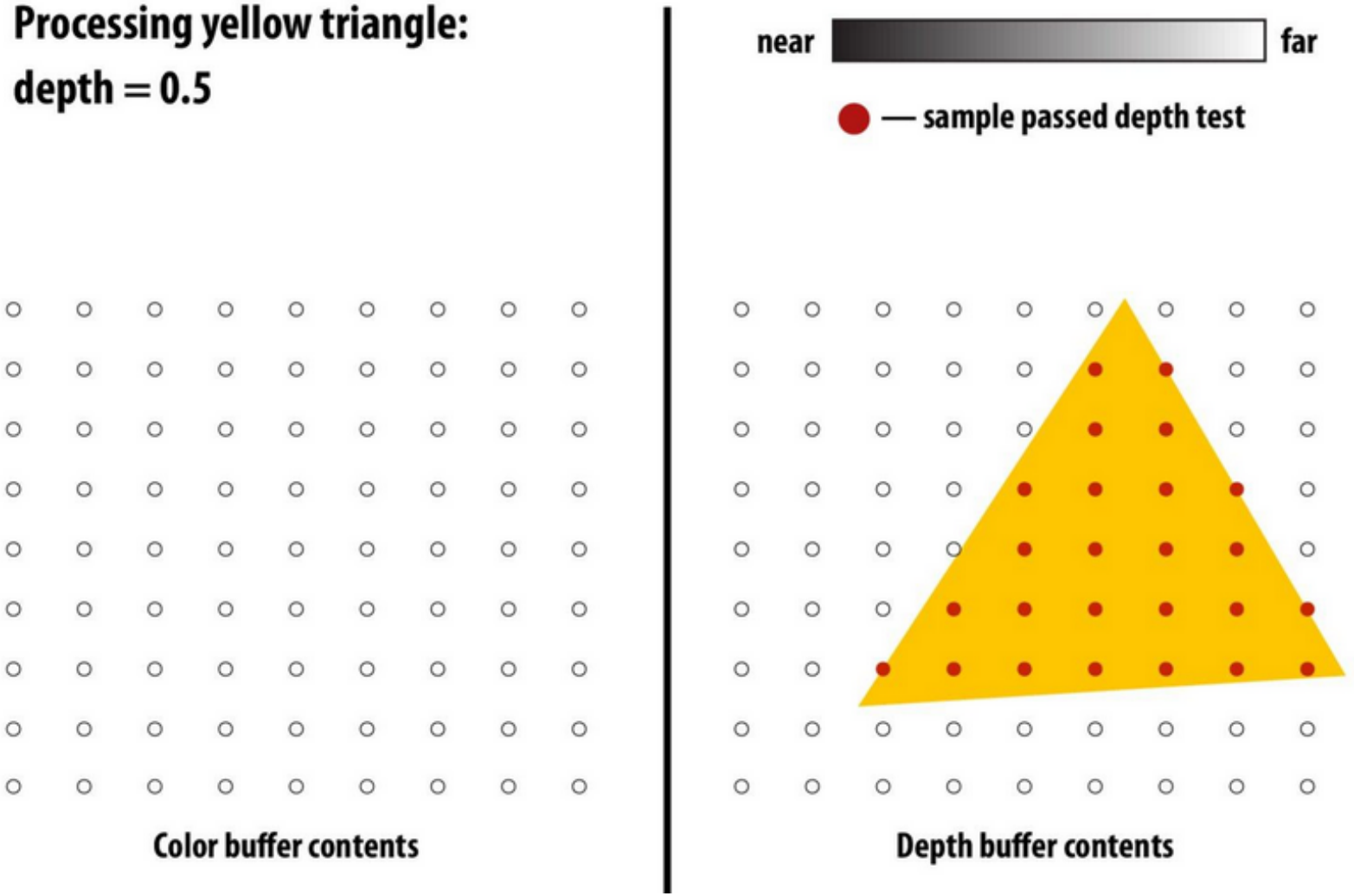
Preuzeto sa: http://15462.courses.cs.cmu.edu/spring2021/lecture/alpha

Primer: renderovanje tri neprozirna trougla

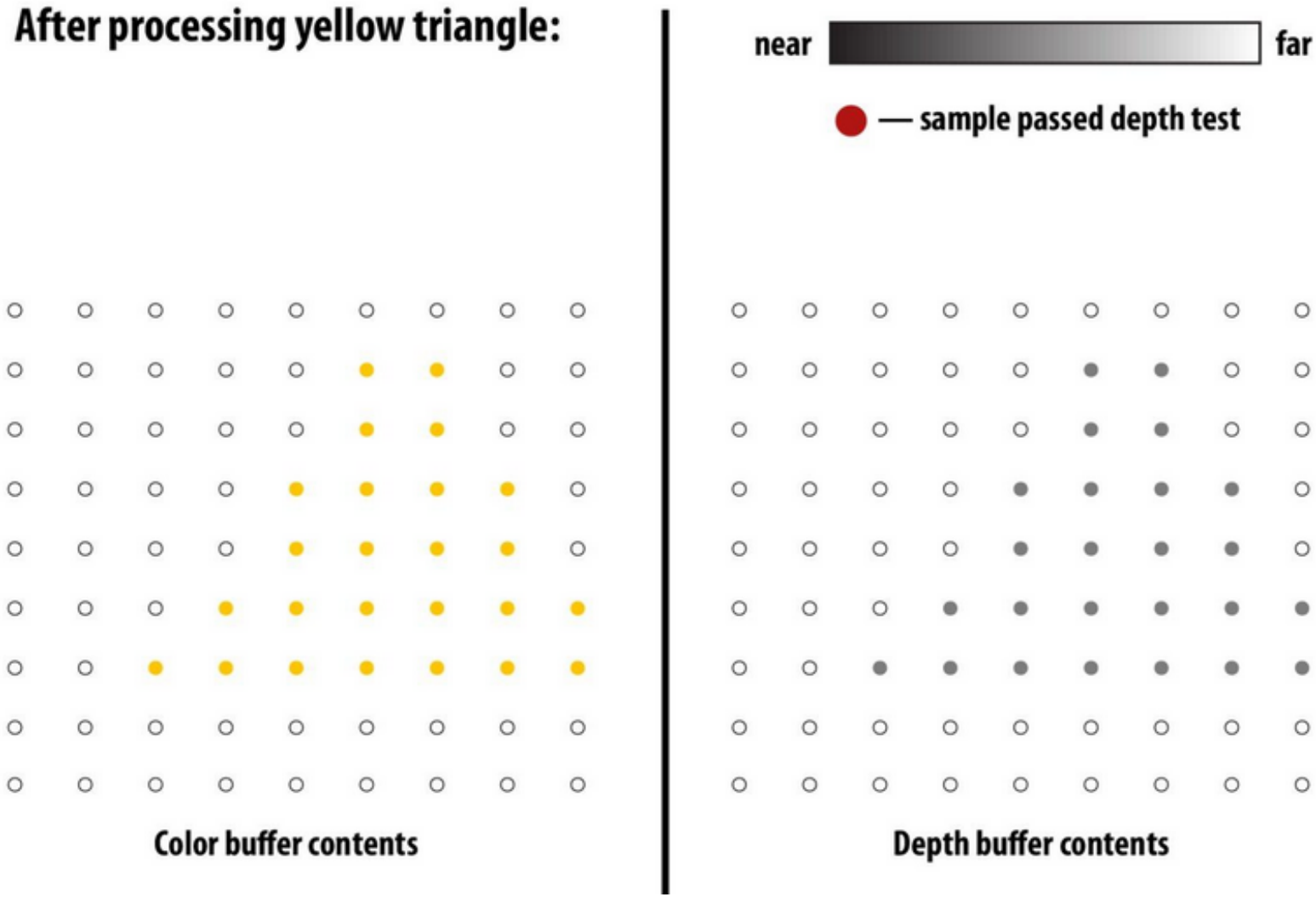
Vrednosti u z-baferu inicijalizujemo na belu



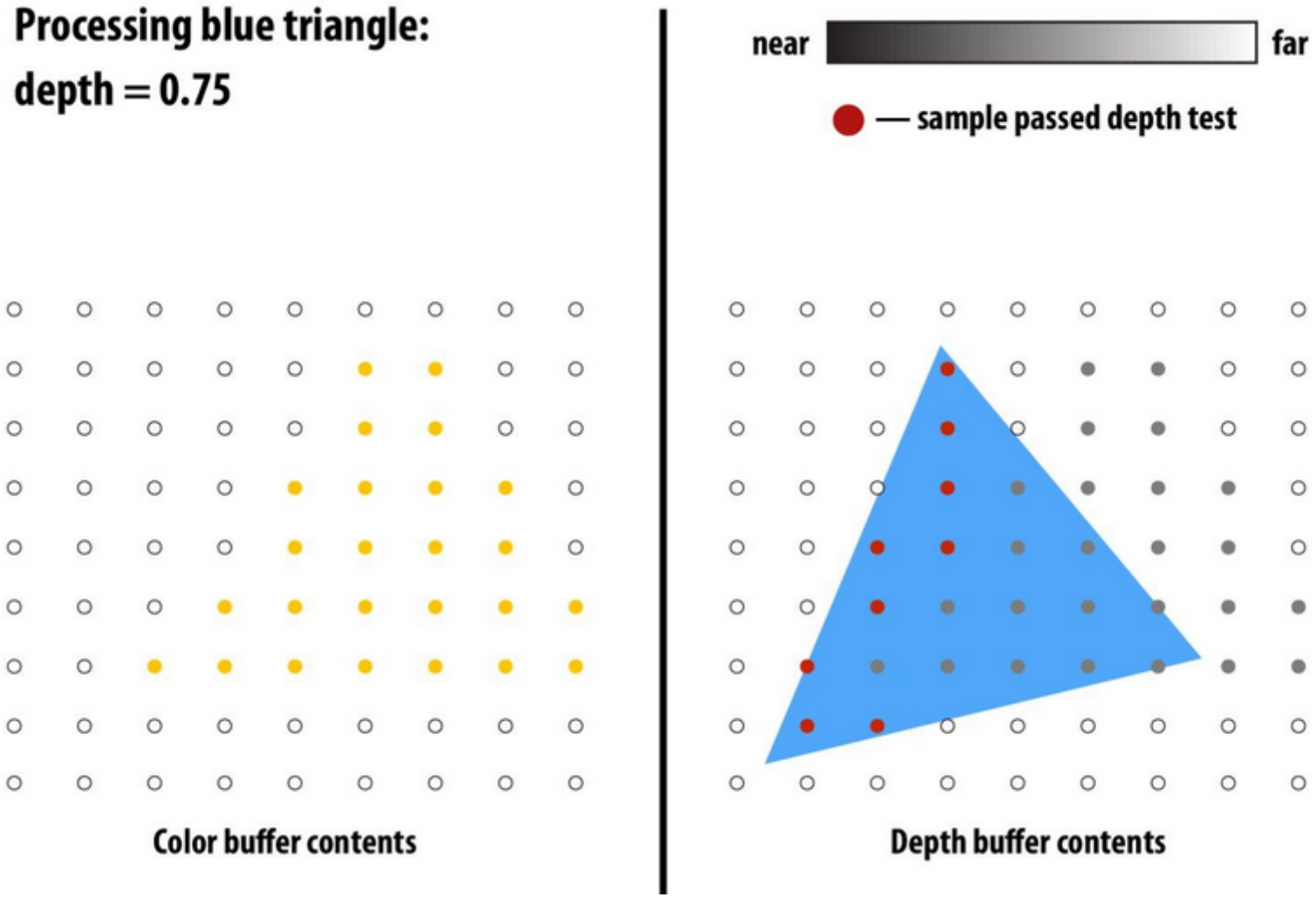
Primer: renderovanje tri neprozirna trougla



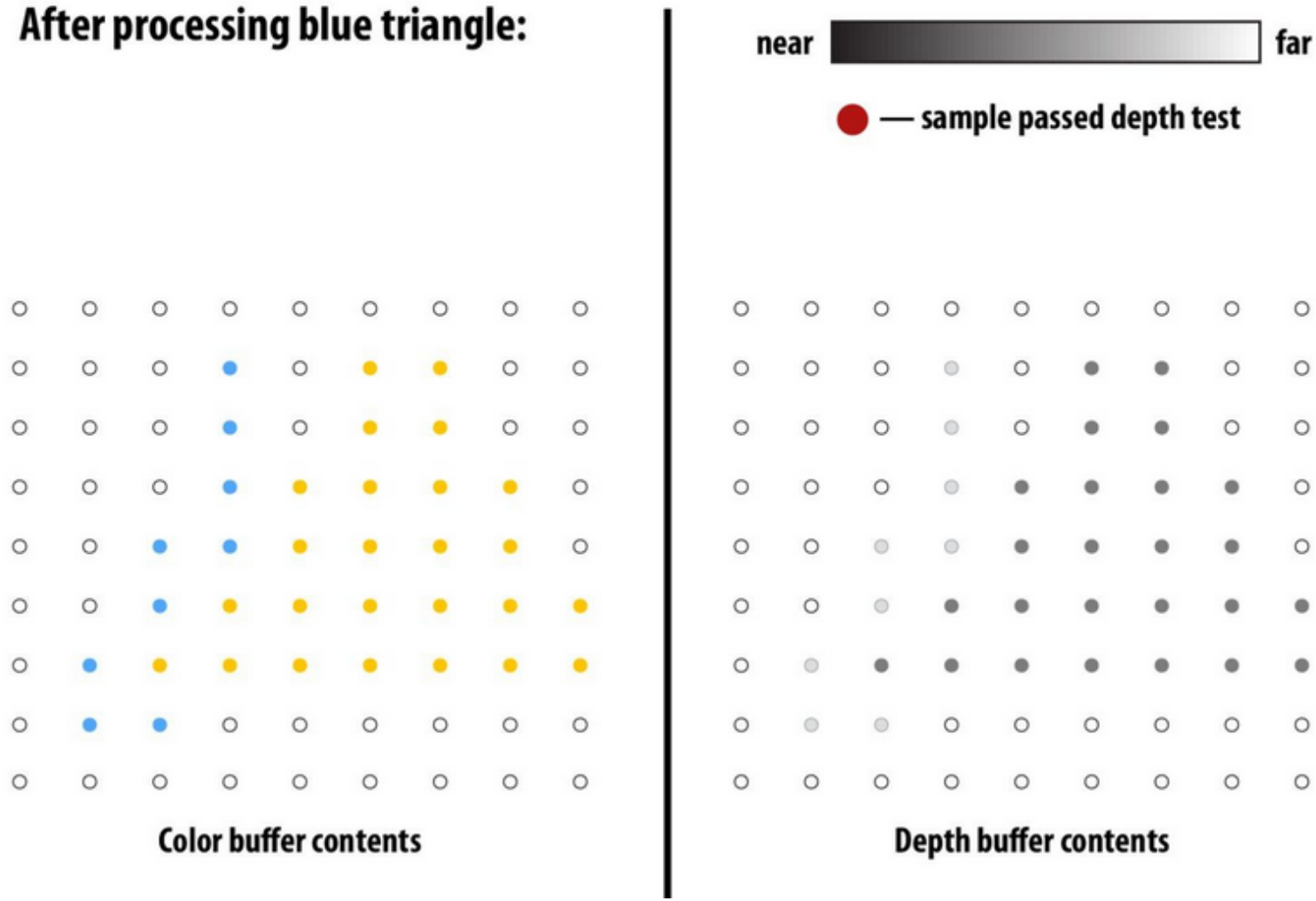
Primer: renderovanje tri neprozirna trougla



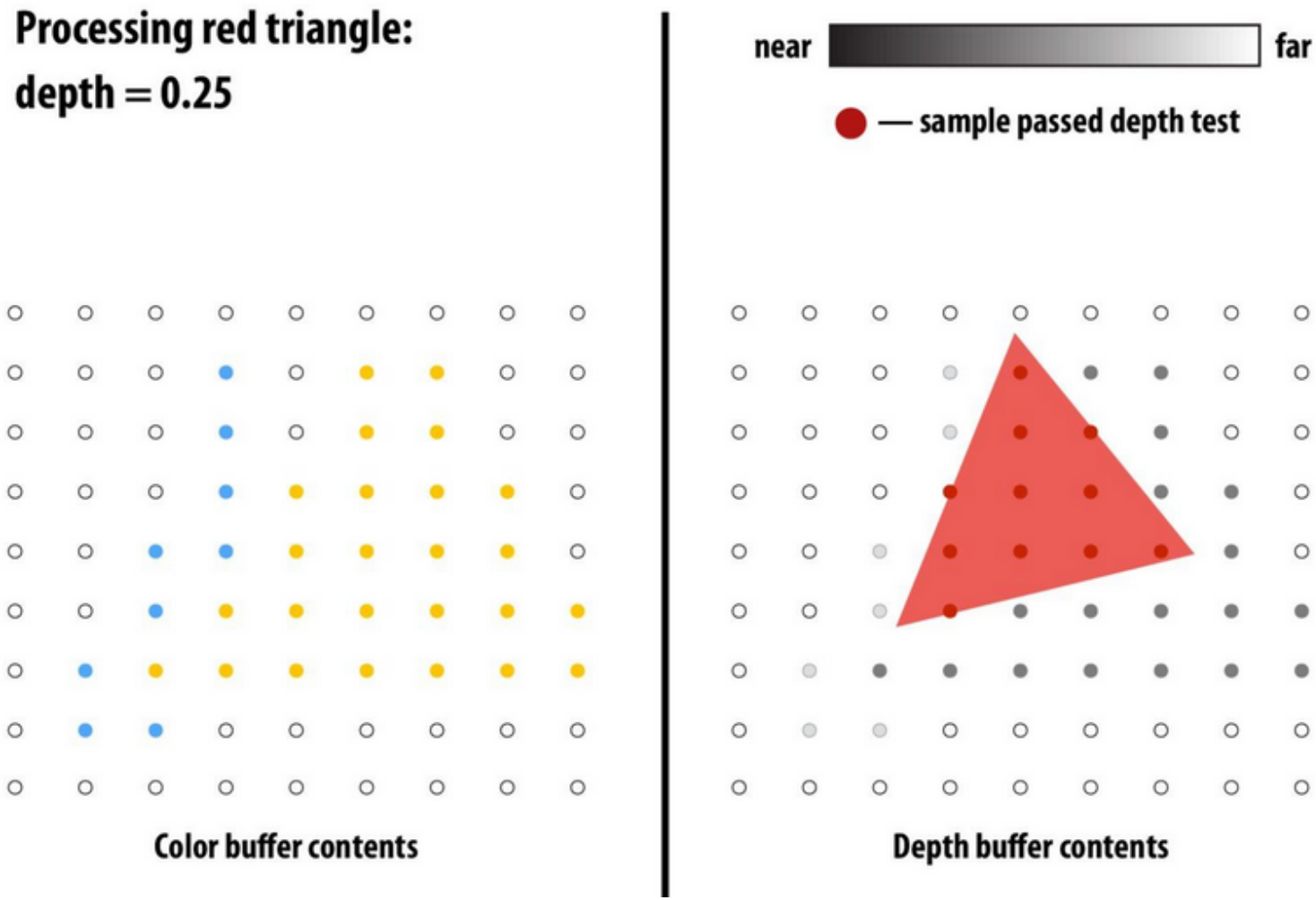
Primer: renderovanje tri neprozirna trougla



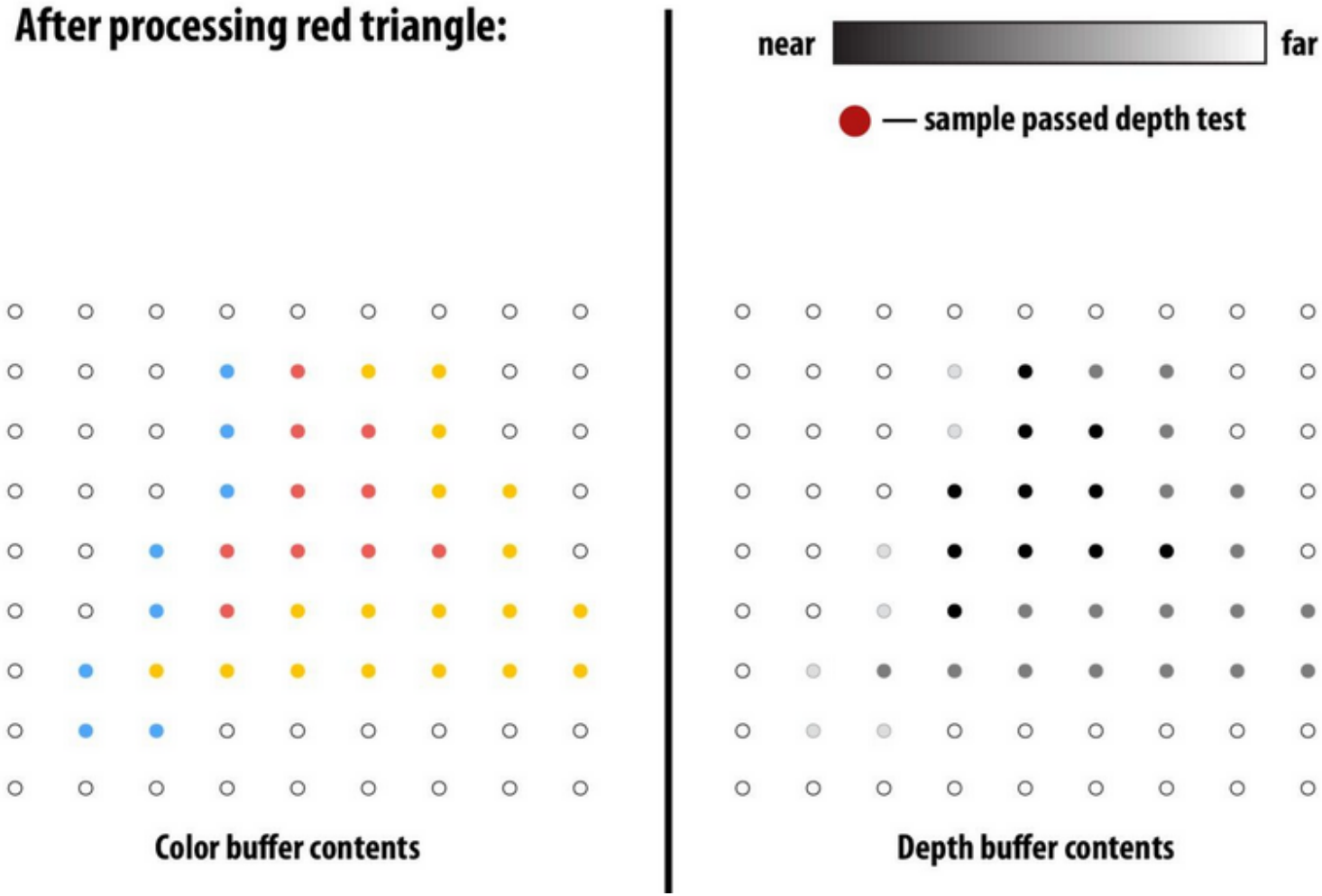
Primer: renderovanje tri neprozirna trougla



Primer: renderovanje tri neprozirna trougla



Primer: renderovanje tri neprozirna trougla

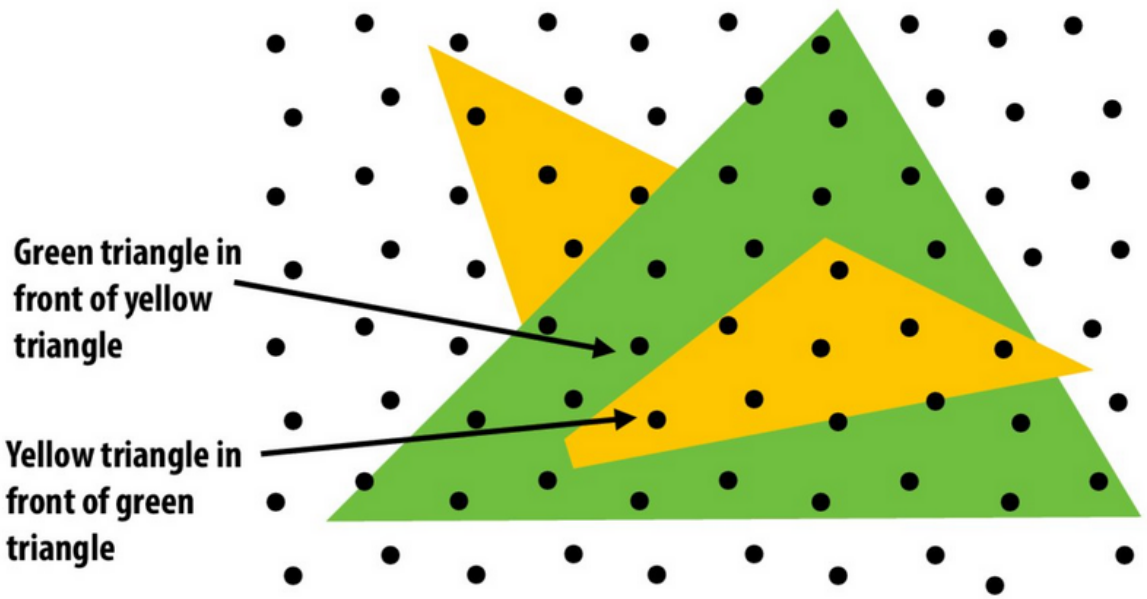


Dubina + preseci primitiva

- Do sad smo razmatrali trouglove koji su u potpunosti jedan ispred drugog
- Da li z-bafer algoritam podržava površi koje se presecaju?

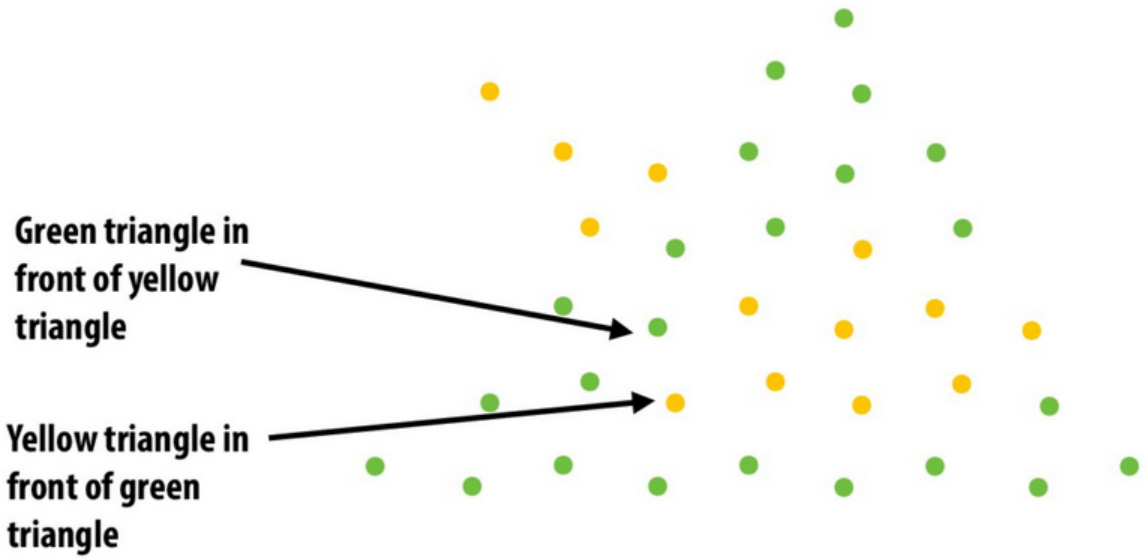
Dubina + preseci primitiva

- Do sad smo razmatrali trouglove koji su u potpunosti jedan ispred drugog
- Da li z-bafer algoritam podržava površi koje se presecaju?
- Test da li jedna primitiva zaklanja drugu se zasniva na dubini trougla u tački u kojoj vršimo uzorkovanje
- Relativna dubina trouglova može biti različita u različitim tačkama trouglova



Dubina + preseci primitiva

- Do sad smo razmatrali trouglove koji su u potpunosti jedan ispred drugog
- Da li z-bafer algoritam podržava površi koje se presecaju?
- Test da li jedna primitiva zaklanja drugu se zasniva na dubini trougla u tački u kojoj vršimo uzorkovanje
- Relativna dubina trouglova može biti različita u različitim tačkama trouglova

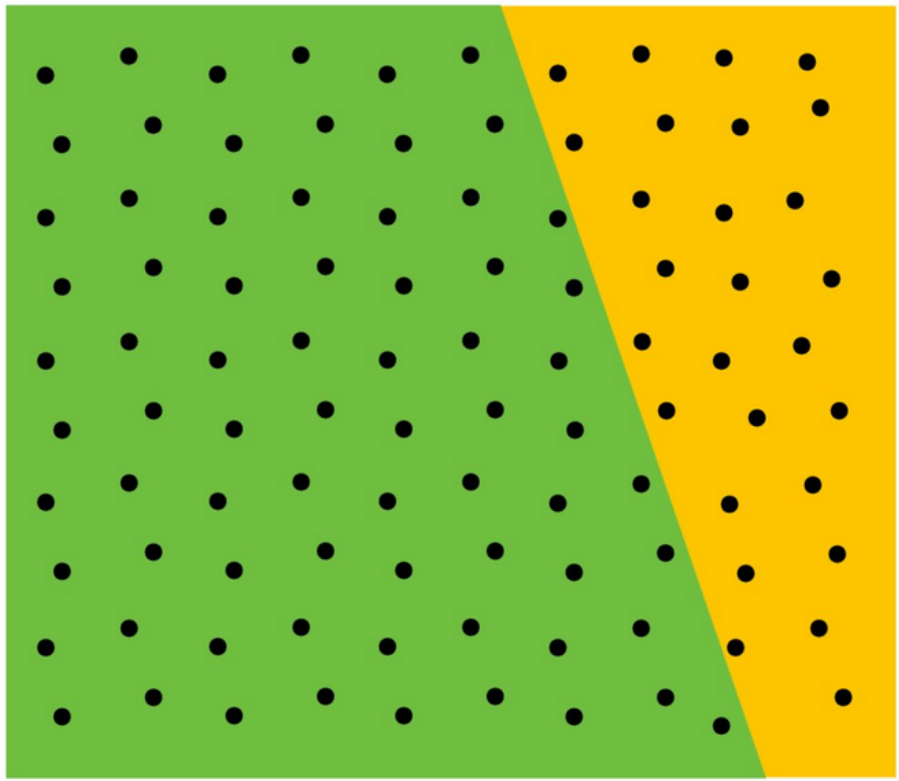


Dubina + nadsemplovanje

- Da li z-bafer podržava nadsemplovanje?

Dubina + nadsemplovanje

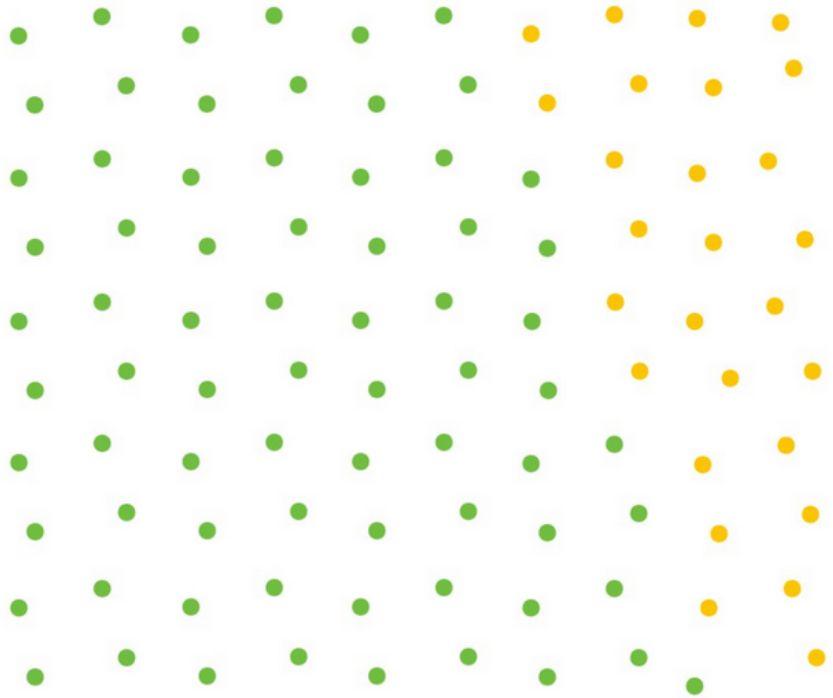
- Da li z-bafer podržava nadsemplovanje?
- Da! Moguće je u z-baferu čuvati vrednosti dubina većeg broja uzoraka po pikselu



(Here: green triangle occludes yellow triangle)

Dubina + nadsemplovanje

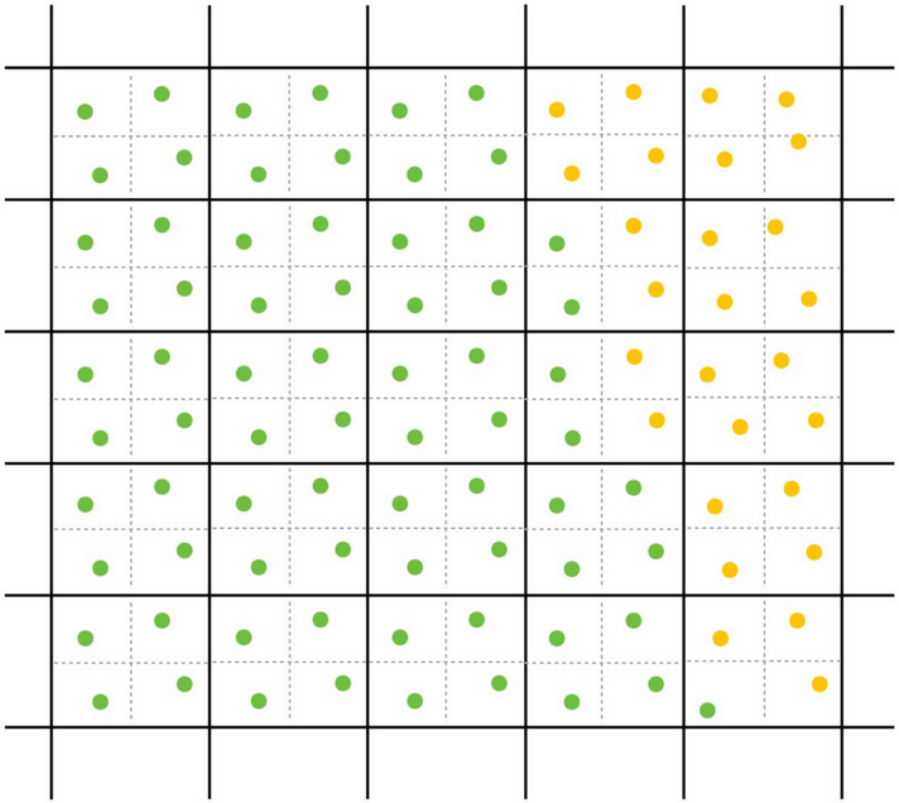
- Da li z-bafer podržava nadsemplovanje?
- Moguće je u z-baferu čuvati vrednosti dubina većeg broja uzoraka po pikselu



Color of super samples after rasterizing w/ depth buffer

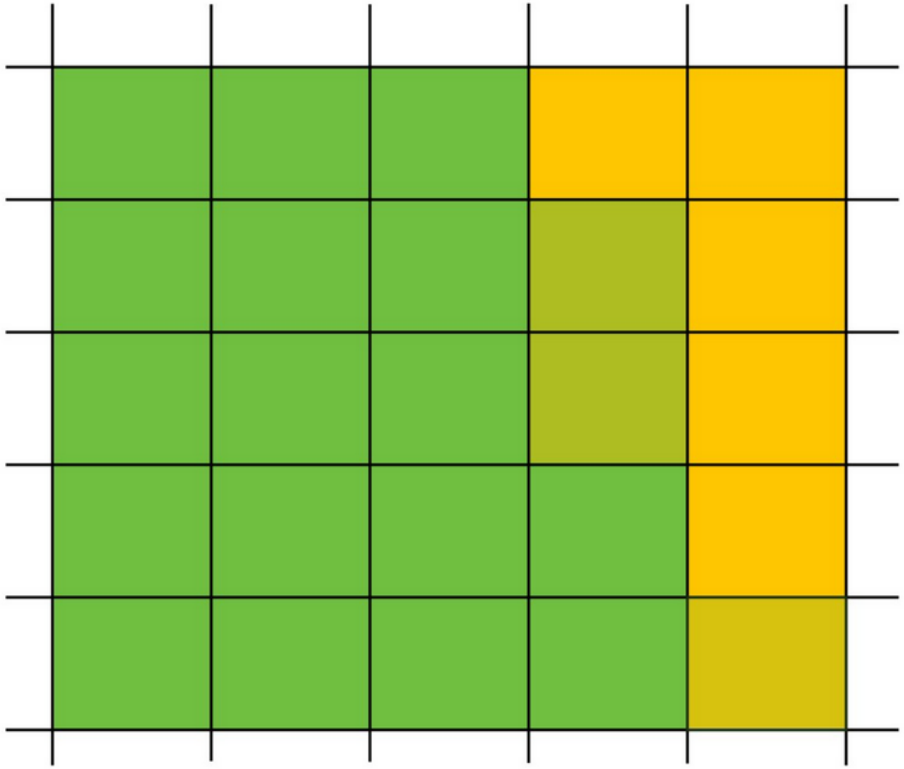
Sadržaj kolor bafera

- Kolor bafer može da sadrži jedan ili veći broj uzoraka boje po pikselu
- Primer kada imamo 4 uzorka po pikselu



Krajnji rezultat

- Napravimo prosek vrednosti uzoraka po pikselu
- Dobijamo efekat glatkog prelaza iz levog poligona u desni



Rezime z-bafer algoritma

- Pamtimo jednu vrednost dubine po uzorku, a ne po pikselu
- Algoritam koristi konstantan dodatni prostor po uzorku, ne zavisi od broja primitiva koje se preklapaju
- Test ispitivanja dubine je konstantne složenosti po uzorku
- Nije potrebno sortiranje objekata pre primene algoritma
- z-bafer algoritam nije ograničen na trouglove: jedino je važno moći izračunati dubinu površi u tački u kojoj vršimo uzorkovanje
- z-bafer algoritam može biti implementiran u hardveru

Slabosti z-bafer algoritma

- Vrednosti u z-baferu su predstavljene konačnim brojem bitova (npr. 8 bitova daje vrednosti iz opsega $\{0, 255\}$)
- Ako dva fragmenta imaju bliske vrednosti dubine, one mogu biti preslikane u istu vrednost u z-baferu i nećemo znati koji je ispred kog
- Da li se ovo dešava u praksi? Da
- Posledica je delovanja matrice transformacije standardne perspektivne u standardnu paralelnu zapreminu pogleda

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

Perspektivna transformacija uzrokuje z-kompresiju

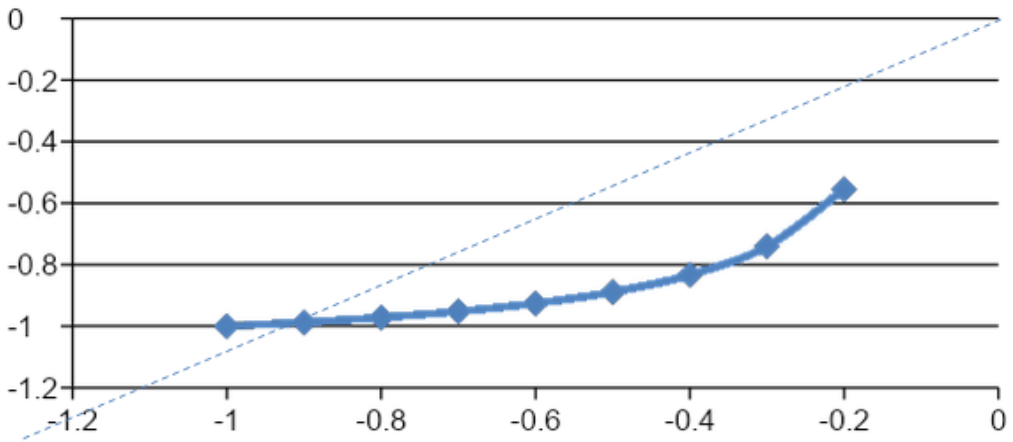
- Pri transformaciji standardne perspektivne u standardnu paralelnu zapreminu pogleda, tačke se pomeraju i kompresuju ka zadnjoj ravni odsecanja

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

- Nakon homogenizacije dobijamo $\begin{bmatrix} -x/z \\ -y/z \\ \frac{c-z}{z+zc} \\ 1 \end{bmatrix}$
- Sa povećanjem z koordinate veći je stepen perspektivnog skraćenja po x i y
- Označimo novu vrednost dubine (tj. z-koordinate) $\frac{c-z}{z+zc}$ sa z'

Perspektivna transformacija uzrokuje z-kompresiju

- Fiksirajmo vrednosti parametara f i n (samim tim i c) i razmotrimo na koji način se menja vrednost za z'
- Npr. za $n = 0.1$ i $f = 1$ važi $c = -0.1$



- Primetimo da se z vrednosti pomeraju i kompresuju ka vrednosti $z' = -1$ u standardnoj paralelnoj zapremini pogleda - ovaj efekat naziva se z-kompresija

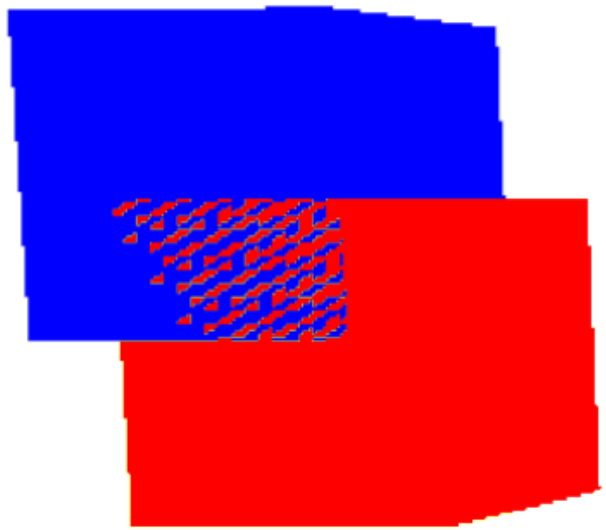
z-kompresija viđena algebarski

- Što je prednja ravan odsecanja bliža ravni $z = 0$, a zadnja ravan odsecanja udaljenija, vrednost $c = -n/f$ je bliskija vrednosti 0, a efekat z-kompresije je izraženiji

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

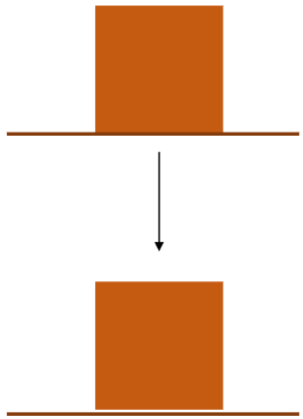
Vizualni efekat z-kompresije: z-konflikt

- z-konflikt se javlja kada dve primitive imaju slične vrednosti u z-baferu
  - komplanarni poligoni
  - jedan se na proizvoljni način bira da bude iznad drugog, ali vrednost z-koordinate varira duž poligona i javlja se ovaj efekat (`https://www.youtube.com/watch?v=CjckWVwd2ek`)
  - efekat jači kada su objekti na većem rastojanju jer je za veće vrednosti z-bafer ima manju preciznost



Kako se boriti sa z-konfliktom?

- Izbegavati da se objekti postavljaju preblizu jedan drugome tako da se neki njihovi trouglovi preklapaju
- Napraviti mali pomak koji je za korisnika jedva primetan



- Povećati preciznost bafera dubine (npr. koristiti 32-bitni z-bafer koji čuva realne brojeve) po cenu
  povećanja memorijskih zahteva – there is no free lunch
- Povećati vrednost razlomka n/f
  - zadnju ravan odsecanja primaknuti posmatraču
  - prednju ravan odsecanja odmaknuti od posmatrača

Transparentni poligoni

- Primetimo da stvari postaju komplikovanije ako su trouglovi transparentni