Računarska grafika

predavanja
v.prof.dr. Samir Lemeš
slemes@unze.ba

22. Algoritmi isijecanja

- Pojam isijecanja
- Isijecanje tačke
- Isijecanje linije
- Algoritam Cohen-Sutherland
- Algoritam Liang-Barsky
- Algoritam Cyrus–Beck
- Algoritam Nicholl-Lee-Nicholl

Pojam isijecanja

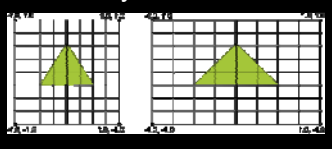
- Isijecanje (*clipping*) je svaka procedura koja uklanja dijelove slike.
- Algoritmi isijecanja se koriste u 2D pogledima da bi se identifikovao dio slike koji se nalazi unutar prozora isijecanja (vidljivi dio slike).
- Matrice transformacija se primjenjuju na isječeni dio slike, kako bi se smanjio obim proračuna

Pojam isijecanja

- Najčešće se koriste sljedeći 2D algoritmi isijecanja:
  - isijecanje tačaka
  - isijecanje linija (pravolnijskih segmenata)
  - isijecanje zatvorenih kontura (poligona)
  - isijecanje krivulja
  - isijecanje teksta

Pojam isijecanja

- Prozor isijecanja se definiše koordinatama krajnjih dijagonalnih tačaka prozora: (X_{min}, Y_{min}) i (X_{max}, Y_{max})
- Tako definisane koordinate odgovaraju normalizovanom kvadratu, gdje se vrijednosti koordinata x i y kreću od 0 do 1 ili od -1 do 1



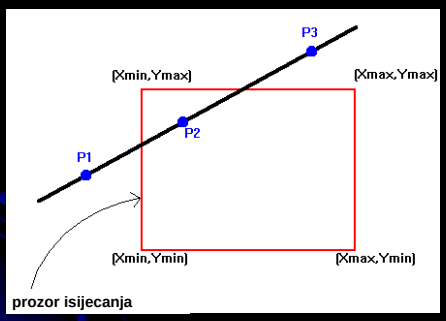
Isijecanje tačke

- Ako su x koordinate okvira isijecanja Xmin i Xmax, a y koordinate Ymin i Ymax, onda sljedeće nejednakosti moraju biti zadovoljene da bi tačka (X,Y) bila unutar okvira isijecanja:

  Xmin < X < Xmax
  i Ymin < Y < Ymax

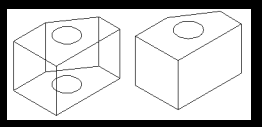
- Ako barem jedna od ove 4 nejednakosti nije zadovoljena, tačka je izvan okvira isijecanja.

Isijecanje tačke

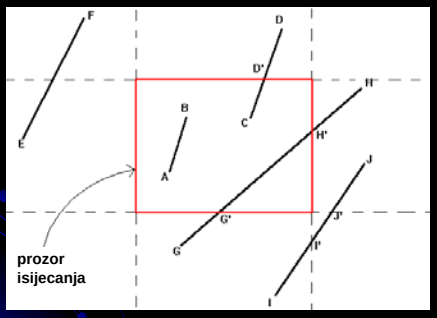


Isijecanje linije

- Kod isijecanja linije, ne posmatraju se sve tačke koje joj pripadaju, nego samo krajnje tačke.
- Ako su obje krajnje tačke unutar okvira isijecanja, cijela linija je vidljiva (trivijalno).
- Ako je jedna krajnja tačka unutar a druga izvan, onda linija siječe okvir i treba izračunati tačku presjeka.
- Ako su obje krajnje tačke izvan okvira, potrebni su dodatni proračuni da bi se utvrdilo da li je dio linije vidljiv.



Isijecanje linije



Isijecanje linije

- Za netrivijalne slučajeve razvijeni su algoritmi za isijecanje linije:
  - Algoritam Cohen-Sutherland
  - Algoritam Liang-Barsky
  - Algoritam Nicholl-Lee-Nicholl

Algoritam Cohen-Sutherland

1. Parovi krajnjih tačaka se prvo provjere da li se trivijalno odbacuju ili prihvataju pomoću binarnih regiona
2. Ako se tako ne može utvrditi vidljivost linije, linija se dijeli na dva segmenta po ivici okvira
3. Iterativno se testiraju segmenti dok se ne dođe do segmenta koji je cijeli vidljiv ili se trivijalno odbacuje.

Algoritam Cohen-Sutherland

Binarni regioni

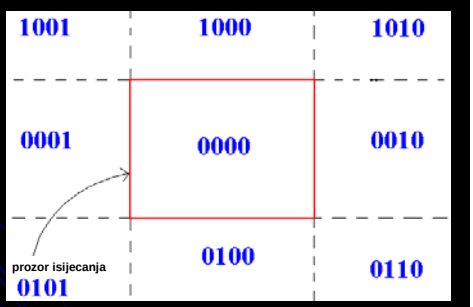
ABCD

gore

dole

desno

lijevo



Algoritam Cohen-Sutherland

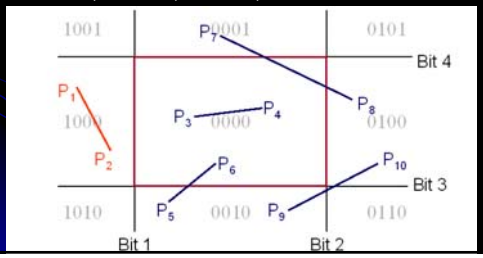
- Bit 1: iznad gornjeg ruba
  Y > Ymax
- Bit 2: ispod donjeg ruba
  Y < Ymin
- Bit 3: desno od desnog ruba
  X > Xmax
- Bit 4: lijevo od lijevog ruba
  X < Xmin

Algoritam Cohen-Sutherland

- Trivijalna rješenja se dobiju logičkim operacijama sa bitovima koji predstavljaju regione.
- Ako su obje krajnje tačke vidljive (OR krajnjih tačaka == 0000): trivijalno vidljivo.
- Ako su obje krajnje tačke u istom dijelu, koji je izvan okvira (AND krajnjih tačaka != 0000): trivijalno nevidljivo.

Zadatak

- Izvršiti klasifikaciju linija koristeći AND (nevidljivo) i OR (vidljivo).
  0: 0 i 0; 0 i 1; 1 i 0; 0 ili 0



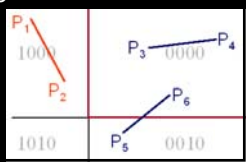
Zadatak

- vidljivo: "ili" mora biti 0000
- nevidljivo: "i" nije 0000

- 1000 ili 1000 = 1000 - ?
- 1000 i 1000 = 1000 – nevidljivo

- 0000 ili 0000 = 0000 – vidljivo
- 0000 i 0000 = 0000 - ?

- 0010 ili 0000 = 0010 - ?
- 0010 i 0000 = 0000 - ?



Algoritam Cohen-Sutherland

- Ako su obje krajnje tačke u različitim regionima, algoritam traži jednu od tačaka koja je izvan okvira.
- Zatim se izračunava tačka presjeka te vanjske tačke i pravca koji prolazi kroz ivicu okvira (parametarskom jednačinom tog pravca).
- Tako izračunata tačka presjeka se uzima kao nova krajnja tačka.
- Algoritam se ponavlja dok se ne dođe do trivijalnog rješenja.

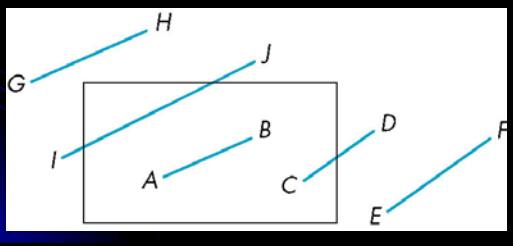
Algoritam Cohen-Sutherland

- Primjer implementacije (Java):
- http://min.nl/cs426/jar/clip.html
- Primjer C++ koda:
- http://www.pracspedia.com/CG/cohen-sutherland-line-clipping.html
- Primjer OpenGL koda:
- http://hubpages.com/art/cohen-sutherland-implementation-using-OpenGL

Algoritam Cohen-Sutherland

- Koji su kodovi regiona za krajnje tačke ovih linija:

1001	1000	1010
0001	0000	0010
0101	0100	0110



Algoritam Liang-Barsky

- Algoritam Liang-Barsky koristi parametarsku jednačinu linije i nejednakosti koje opisuju granice okvira isijecanja da bi se odredili presjeci između linije i okvira isijecanja.
- Pomoću tih presjeka se može znati koji dio linije je vidljiv
- Algoritam je efikasniji od algoritma Cohen-Sutherland (36% za 2D linije, 40% za 3D linije, 70% za 4D linije).

Algoritam Liang-Barsky

1. Predstaviti segmente linije u parametarskoj formi
2. Izvesti jednačine za testiranje da li je tačka unutar prozora
3. Izračunati nove vrijednosti parametara za vidljivi dio segmenta, ako takav postoji
4. Prikazati vidljivi dio segmenta linije

Algoritam Liang-Barsky

- Prikaz segmenta linije u parametarskom obliku:
  - $x = x_1 + (x_2 - x_1) * m = x_1 + dx * m$ $0.0 < m < 1.0$
  - $y = y_1 + (y_2 - y_1) * m = y_1 + dy * m$
- Kad je m = 0.0 => x_1, y_1
- Kad je m = 1.0 => x_2, y_2

Algoritam Liang-Barsky

- Određuju se tačke $P_i * m < q_i$ $i = 1, 2, 3, 4$
- gdje je:

$P_1 = -dx$	$q_1 = x_1 - X_{min}$	Lijevo
$P_2 = dx$	$q_2 = X_{max} - x_1$	Desno
$P_3 = -dy$	$q_3 = y_1 - Y_{min}$	Dole
$P_4 = dy$	$q_4 = Y_{max} - y_1$	Gore

Algoritam Liang-Barsky

- Vidljivi dio linije počinje od najveće vrijednosti m: $m_1 = MAX (\{q_i / P_i \mid P_i < 0, i = 1, 2, 3, 4\} U \{0\})$
- Za $P_i > 0$ dobije se završna tačka vidljivog dijela: $m_2 = MIN (\{q_i / P_i \mid P_i > 0, i = 1, 2, 3, 4\} U \{1\})$
- Ako postoji vidljivi segment, on odgovara parametarskom intervalu $m_1 <= m <= m_2$ i $m_1 <= m_2$
- Ako je $m_1 > m_2$ linija se cijela ne vidi, a ako nije treba izračunati krajnje tačke od m_1, m_2.

Algoritam Cyrus–Beck

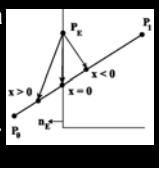
- Sličan algoritmu Liang-Barsky.
- Razlika je u tome što je Liang-Barsky pojednostavljena varijacija algoritma Cyrus-Beck koja je optimizirana za pravougaoni prozor isijecanja.
- Algoritam Cyrus-Beck je primarno namijenjen za isijecanje linije u parametarskoj formi konveksnim poligonom u 2 dimenzije, odnosno konveksnim poliedrom u 3D.

Algoritam Cyrus–Beck

- Dizajniran je radi povećanja efikasnosti algoritma Sutherland-Cohen, koji koristi ponavljanje isijecanja.
- Za razliku od algoritma Sutherland-Cohen, koji se koristi samo za prozore isijecanja oblika pravougaonika, algoritam Cyrus-Beck se može koristiti s prozorom isijecanja u obliku konveksnog poligona.

Algoritam Cyrus–Beck

- Parametarska jednačina linije ($0 \le t \le 1$): $p(t) = tp_1 + (1-t)p_0 = p_0 + t(p_1-p_0)$
- Za određivanje tačke presjeka s prozorom isijecanja koristi se proizvod tačaka.
- Ako je p_E tačka na ravni isijecanja E:
  - $n\cdot(p(t)-p_E) > 0$: vektor prema unutra
  - $n\cdot(p(t)-p_E) = 0$: vektor paralelno
  - $n\cdot(p(t)-p_E) < 0$: vektor prema vani
- Oznaka n je normala na ravan E.

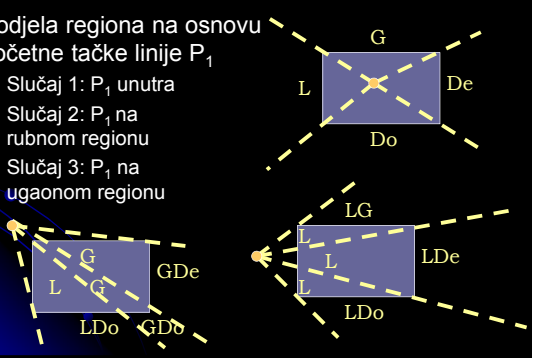


Algoritam Nicholl-Lee-Nicholl

- Najsloženiji algoritam
- Najbrži algoritam
- Radi dobro samo u 2D, za razliku od prethodno prikazanih algoritama koji se mogu primijeniti i na 3D linije
- Na osnovu položaja druge krajnje tačke, dodaje nove regione

Algoritam Nicholl-Lee-Nicholl

- Podjela regiona na osnovu početne tačke linije P_1
  - Slučaj 1: P_1 unutra
  - Slučaj 2: P_1 na rubnom regionu
  - Slučaj 3: P_1 na ugaonom regionu



Algoritam Nicholl-Lee-Nicholl

- Ako je tačka P_1 u jednom od regiona označenih slovom L, onda se linija odsijeca na lijevoj granici okvira isijecanja i čuva se segment linije od te presječne tačke do druge krajnje tačke P_2.
- Ako je krajnja tačka P_2 u regionu LG, onda se čuva segment od lijeve do gornje granice okvira, itd.