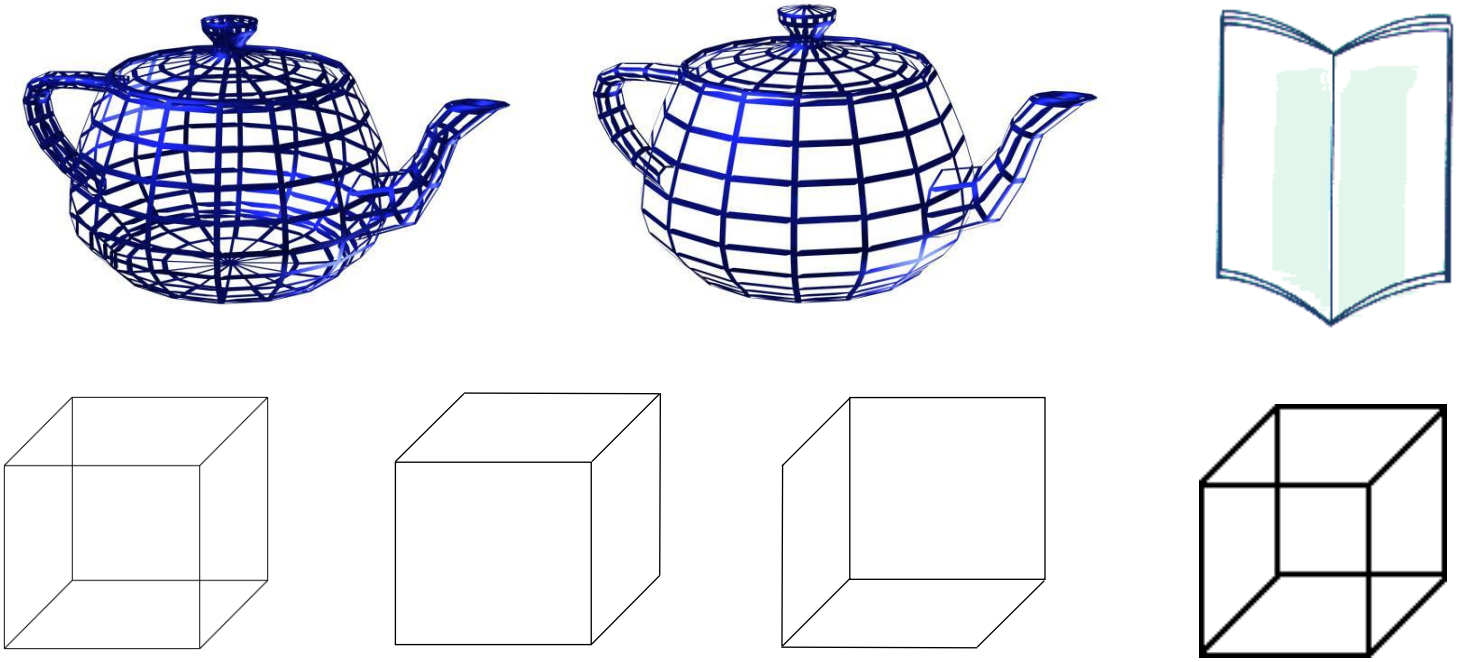
7. Uklanjanje skrivenih linija i površina



Određivanje vidljivosti (engl. hidden-line/surface algorithms)

- Za čovjeka je uklanjanje skrivenih linija i površina jednostavan problem. http://www.echalk.co.uk/amusements/OpticalIllusions/hollowMask/hollowface.html
- Za računalo to je znatan problem, npr.

http://xeogl.org/examples/#effects_ghost

Kod uklanjanja skrivenih linija i površina postoji niz algoritama koji se koriste na raznim mjestima u grafičkom protočnom sustavu.

Neki su brzi i jednostavni neki su složeniji. Vrlo slični algoritmi primjenjivi su i u raznim drugim kontekstima kao što je na primjer ispitivanje sudara objekata ili postupci za određivanje sjene objekta.

U nastavku prezentacije pogledat ćemo:

- određivanje odnosa točke i konkavnog poligona
- brza provjera da li se poligoni/tijela sigurno ne preklapaju (min-maks, brza provjera s opisanim kuglama),
- generiranje prikaza 3D scene (slikarov algoritam, Warnockov postupak, Watkinsov postupak, BSP)
- postupke odsijecanja linija poligona i tijela (Cohen Sutherland, Cyrus-Beck)
- ubrzavanje postupaka ispitivanja sudara (BSP, četvero stabla, oktalna stabla)

Osnove postupaka uklanjanja skrivenih linija i površina

- geometrijska izračunavanja – uspostavljaju odnos između poligona, bridova i točaka (*containment test*)
- geometrijsko uređivanje (*geometric sorting*)
- postupci pretraživanja (*search algorithms*)
- međusoban ovisnost i obilježja (*coherence*)

Određivanje vidljivosti

- uklanjanje objekata izvan piramide pogleda (dijelova poligona)
  https://threejs.org/examples/#webgl_clipping
- uklanjanje stražnjih pogona (engl. *back face culling*)
  http://alteredqualia.com/three/examples/webgl_terrain_dynamic.html
- brzi, jednostavni postupci za rješavanje trivijalnih slučajeva

  (npr. Min-maks provjera)
- utjecaj uklanjanja objekata na scenu (zrcaljenje, sjene) promjena složenosti prikaza ovisno o udaljenosti (engl. LOD-level of detail)
  https://29a.ch/sandbox/2012/terrain/ ToggleWireframe /HighLOD

Podjela postupaka za uklanjanje skrivenih linija i površina

- postupci u prostoru objekta 3D
- postupci u prostoru slike (projekcije) 2D

Slični postupci koriste se kod

- odsijecanja (engl. clipping) https://cedricpinson.github.io/osgjs-website/examples/picking/
- otkrivanje sudara tj. kolizije (engl. collision detection)
- bačene sjene

Geometrijska izračunavanja

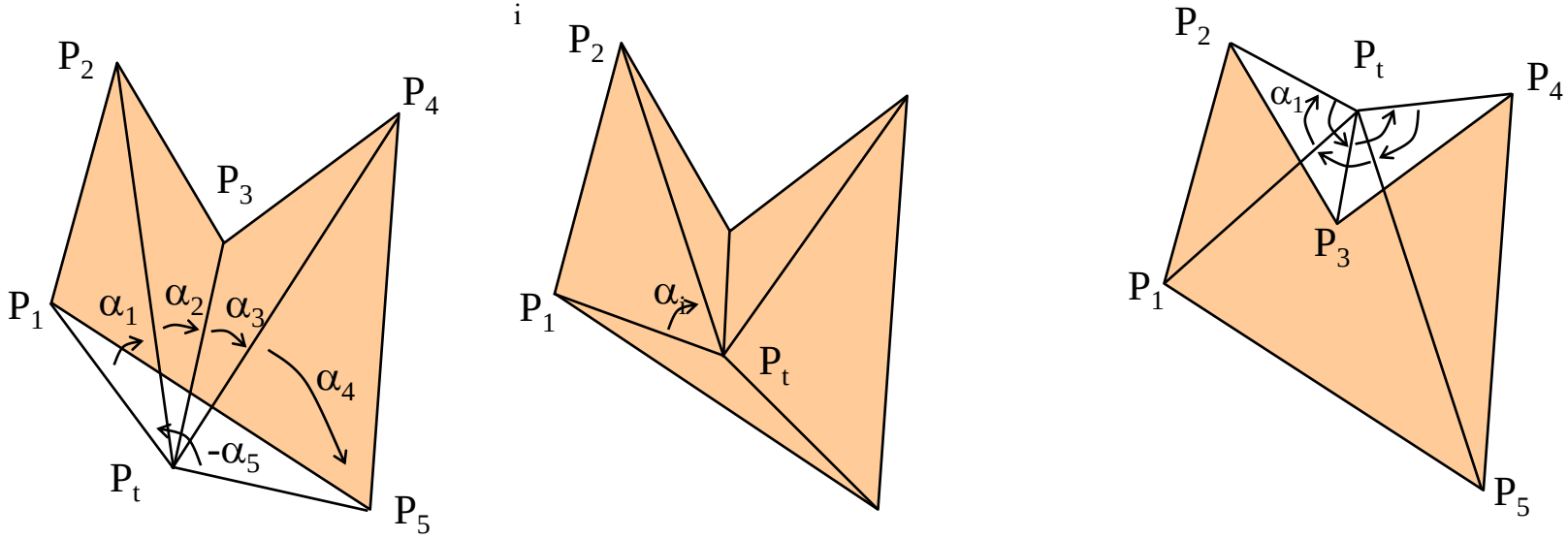
- čine osnovu u postupcima uklanjanja skrivenih linija i poligona, detekcija kolizija, odsijecanja (u prostoru projekcije ili scene).

  - položaj točke prema
    - pravcu ili ravnini
    - poligonu ili tijelu
  - položaj dužine
    - pravcu ili ravnini
    - poligonu ili tijelu
  - Booleove operacije (unija, presjek, razlika)
    - dvaju tijela (engl. solid modelling)
  - određivanje orijentacije poligona (u projekciji)

Provjera odnosa točke i poligona

Korištenje sume kutova. Kut možemo izračunati skalarnim produktom vektora P_1-P_t, P_2-P_t, dok je predznak određen smjerom vektora dobivenog vektorskim produktom istih vektora

- ako je $\sum_i \alpha_i = 0^\circ$ P_t je izvan poligona
- ako je $\sum_i \alpha_i = 2\pi$ P_t je unutar poligona



Za konveksne poligone npr. trokute kojima u OpenGL-u često modeliramo tijela možemo primijeniti i jednostavniji način provjere.

Min-maks provjera

Brzi zahvat koji ustanovljava da li se dva poligona **sigurno ne prekrivaju**.

Neka su poligoni zadani vrhovima $P_1(V_{11} .. V_{1n})$ i $P_2(V_{21} .. V_{2m})$.

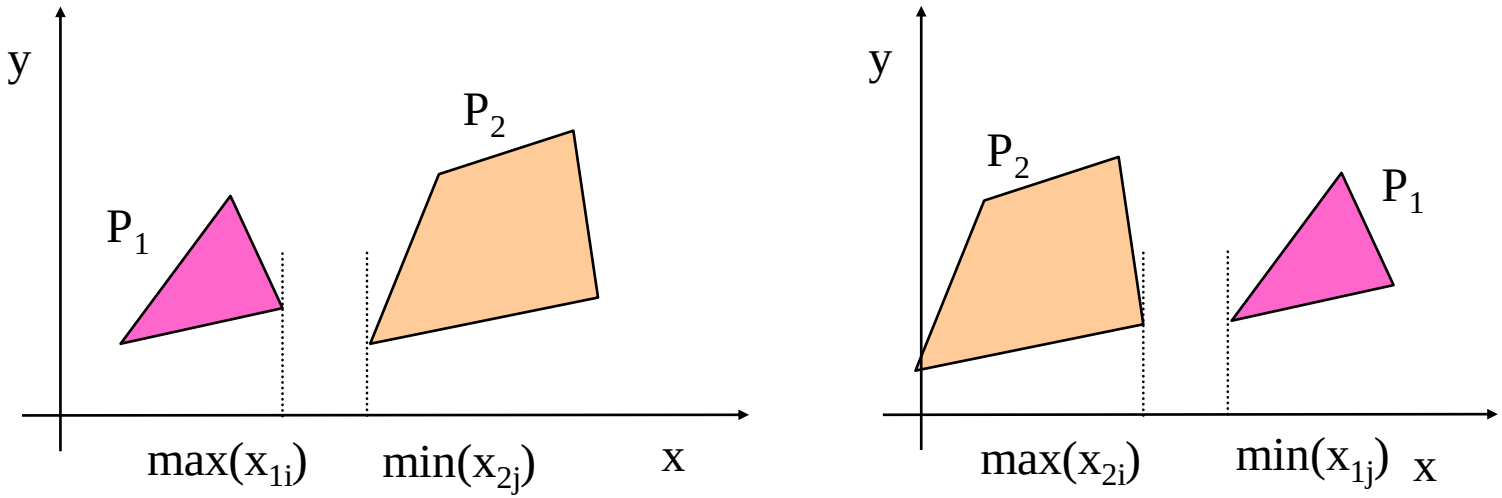
Poligoni se sigurno ne prekrivaju ako vrijedi za svaki i, j:

$\max(x_{1i}) < \min(x_{2j})$ ili

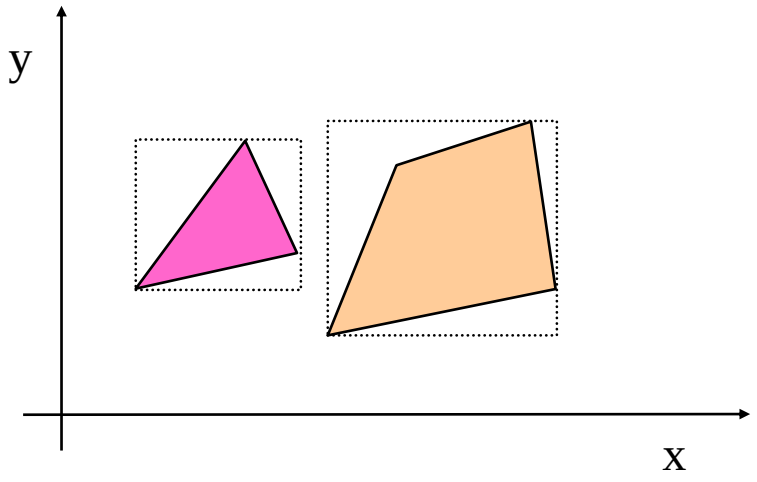
$\max(x_{2i}) < \min(x_{1j})$ ili

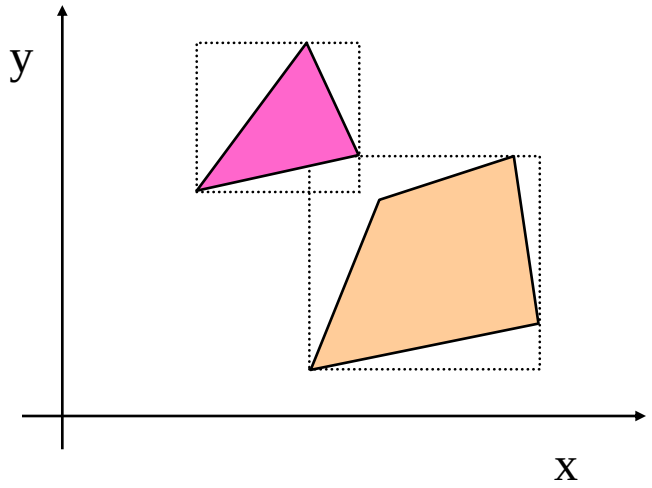
$\max(y_{1i}) < \min(y_{2j})$ ili

$\max(y_{2i}) < \min(y_{1j})$



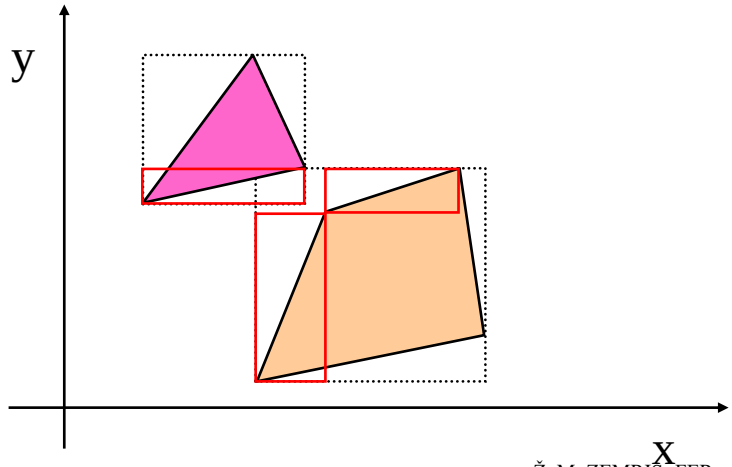
U stvari provjeravamo da li se opisani pravokutnici (engl. *screen extent*, *bounding box*) ne prekrivaju.





Ukoliko se poligoni potencijalno prekrivaju potrebne su daljnje provjere.

Algoritam možemo primijeniti i na bridove.

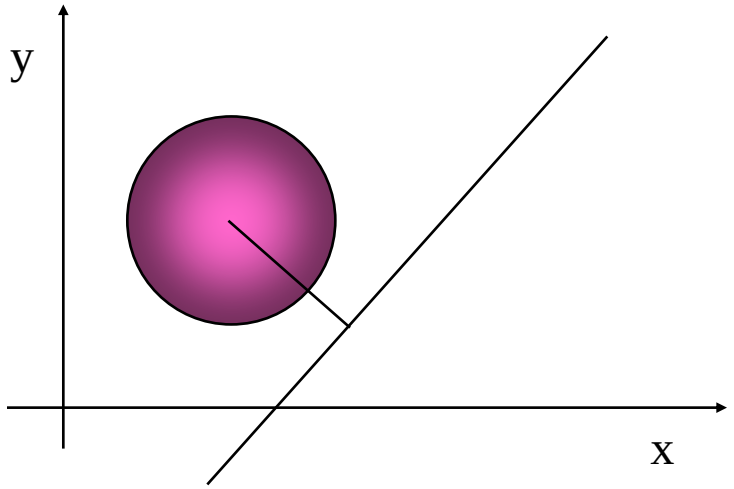


Proširenje algoritma Min-maks na trodimenzjski prostor.

Provjeravamo da li se kvadri (engl. bounding box), koji obuhvaćaju tijela ili dijelove tijela preklapaju. Postupak se obično koristi kod ispitivanja sudara za definiranje sudarača (*collider*).
http://mozdevs.github.io/gamedev-js-3d-aabb/api_box.html Esc

Umjesto kvadara često se koriste i kugle (sfere). Ukoliko je udaljenost pravca do središta kugle:

- veća od polumjera, pravac ne siječe kuglu.
- manja od polumjera, pravac siječe kuglu.



Geometrijsko uređivanje

Sada razmatramo iscrtavanje 3D poligona na 2D prikazu:

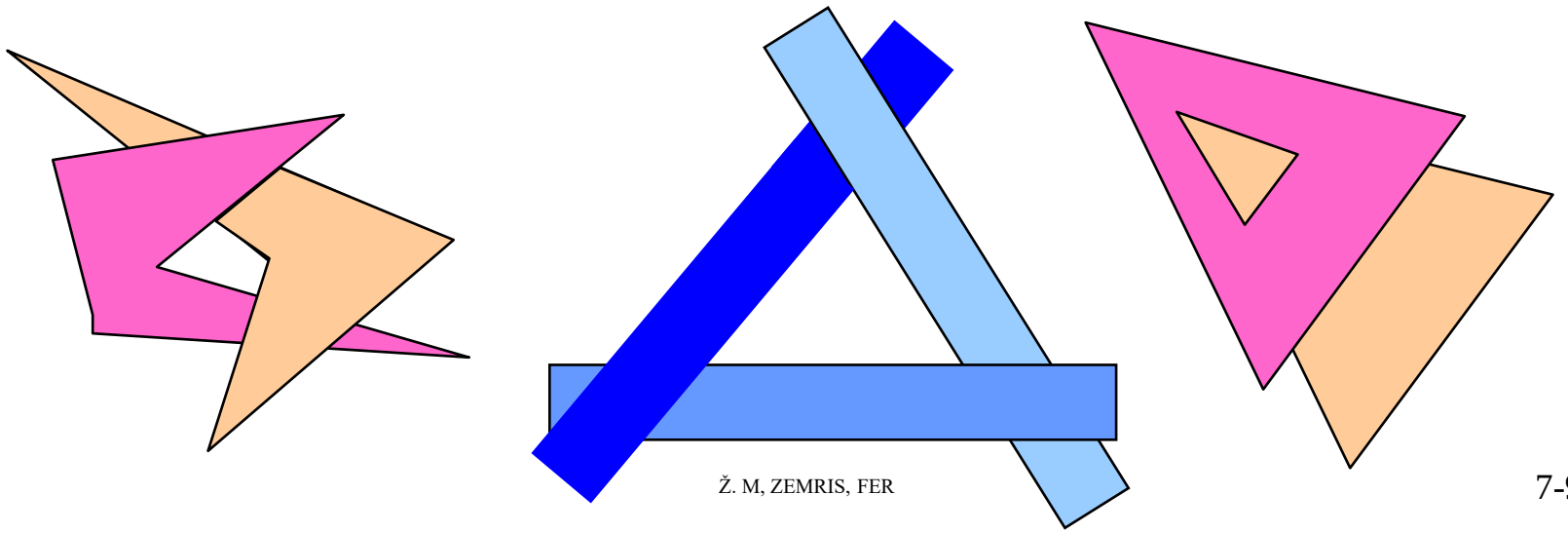
Slikarov algoritam iscrtavanja, BTF (engl. *back to front* to jest *Painter's algorithm*) - nakon iscrtavanja najudaljenijih poligona crtaju se redom sve bliži koji prekrivaju već iscrtane.

Prednosti: - mogućnost korištenja prozirnosti

Nedostaci: - suvišno iscrtavanja prekrivenih poligona

- problem kod iscrtavanja površina ili probadanja
- sortiranje poligona/objekata po udaljenosti

http://www.realtimerendering.com/udacity/?load=demo/unit1-render-mode-1.js



Watkinsov postupak (engl. *scan line method*)

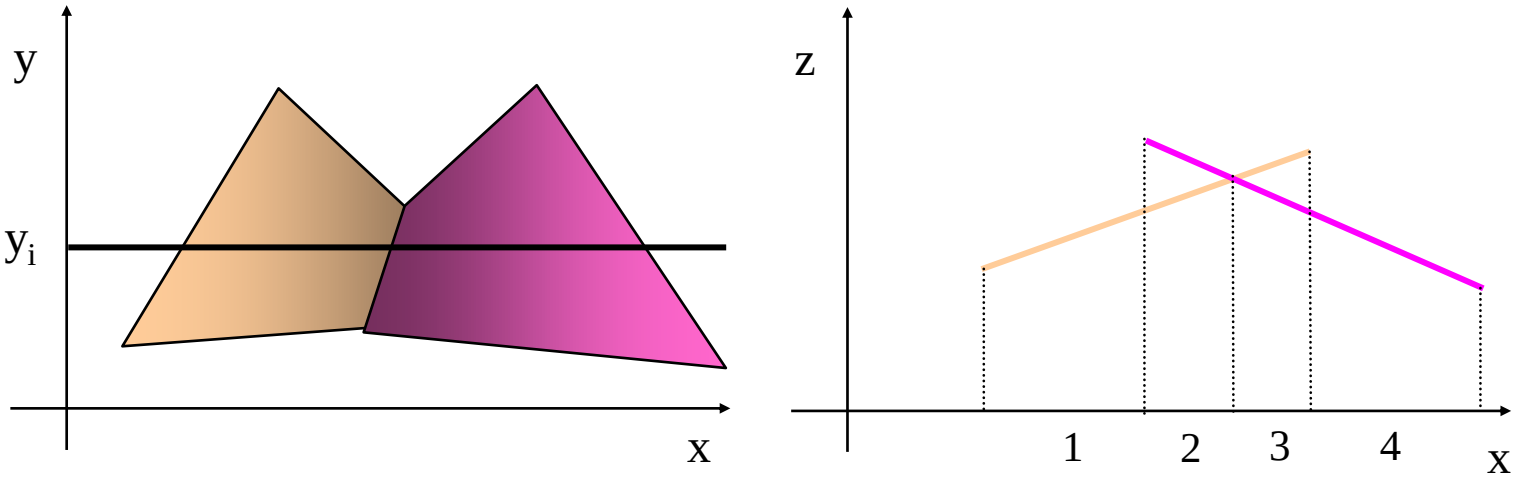
Radi u prostoru slike. Projiciramo poligone u ravninu xy.

U ravnini projekcije postavimo ispitnu liniju.

1. Određivanje raspona uzorka

Raspon uzorka definiramo kao dio linije na kojoj se ne može dogoditi promjena vidljivosti. Raspon uzorka je dio linije koji zadovoljava uvjete:

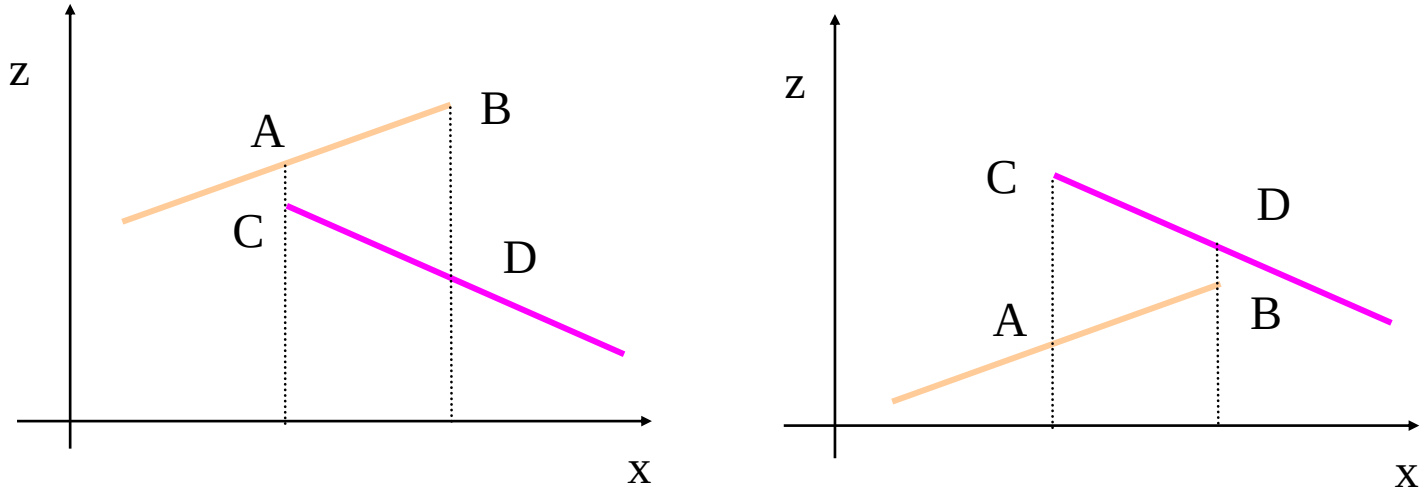
- broj segmenata u rasponu uzorka *konstantan* je i veći od 1
- projekcije presjeka za $y = y_i$ unutar raspona uzorka u ravnini *xz ne sijeku* se unutar raspona uzorka (svaki presjek u projekciji označava novi raspon uzorka)



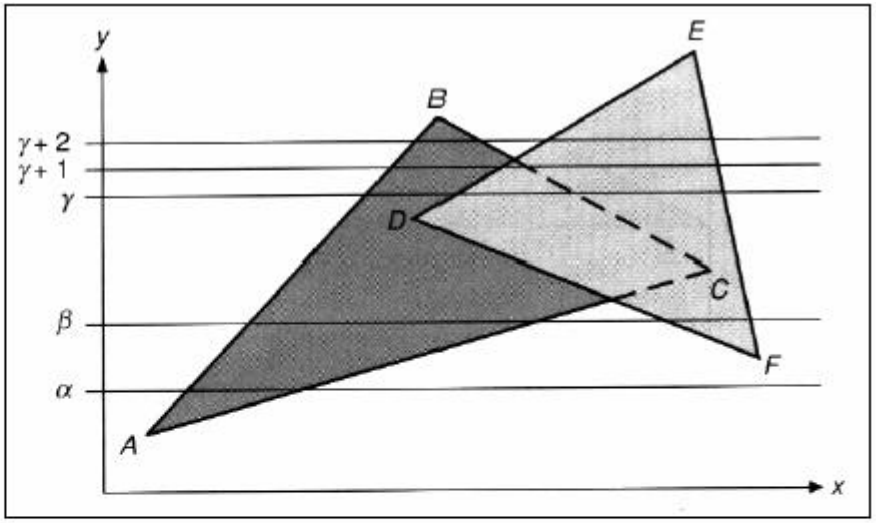
2. Određivanje vidljivosti
3. Raspon uzorka testiramo obzirom na vidljivost:
   - ako je broj segmenata na tekućoj liniji pretrage različit od broja segmenata na prethodnoj liniji pretrage ili
   - ako se krajnje točke dvaju segmenata zamijene, po veličini koordinate z (na primjer: točke A i C te B i D) kad prelazimo iz tekuće u susjednu liniju pretrage (tada se obje površine sijeku u prostoru između dviju ravnina pretrage).

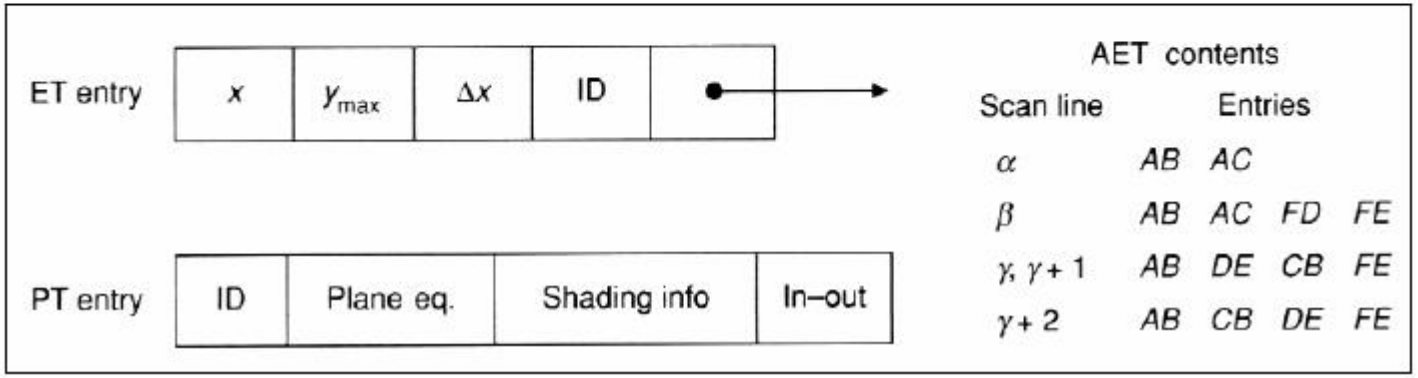
Ispitivanjem z-koordinate možemo odrediti koji segment u rasponu uzorka prekriva druge segmente. Watkins (minilekcije)

(npr: 3D Studio MAX software renderers)



ET - the edge table
PT - the polygon table
AET - the active edge table





ET – x koordinata kraja koji ima manju y koordinatu

- y_{max} y koordinata drugog kraja
- Δx prirast po x (nagib)
- ID poligon kojem brid pripada

https://airtightinteractive.com/demos/js/ruttetra/ sample image

Warnockov postupak

Podjela prostora četverostablom http://paperjs.org/examples/division-raster/

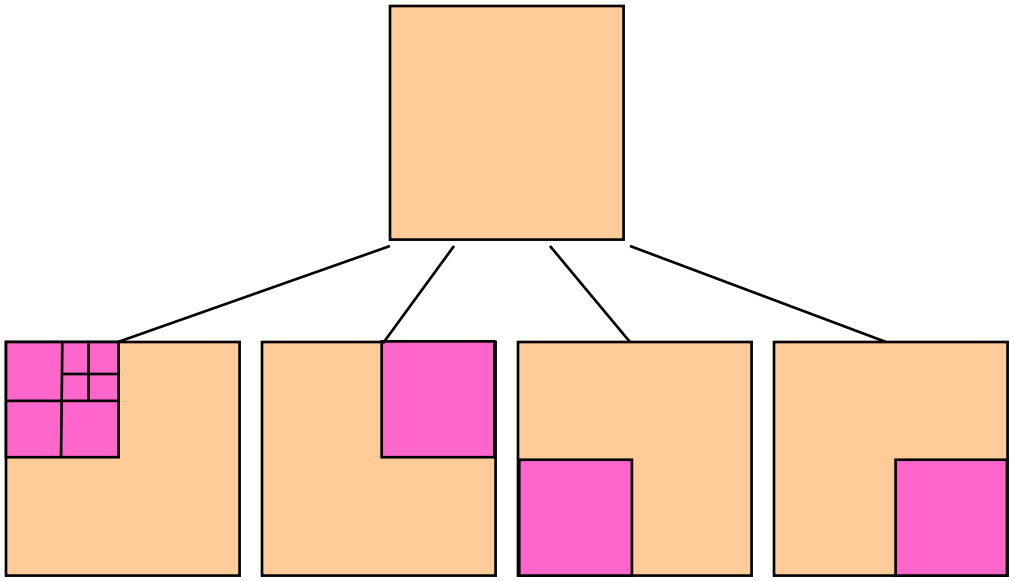
Postupak radi u prostoru slike. Analiziramo sadržaj ispitnog prostora koji je u početku jednak zaslonu – rekurzivna podjela. Pogodan za paralelizaciju.

Mogući slučajevi:

dosegnuli smo maksimalnu unaprijed zadanu dubinu rekurzije (prazno)

(1)-(4) prozor je prazan ili je scena u prozoru jednostavna i moguće ju je prikazati

(0) scena u prozoru je složena, rekurzivno dijelimo dalje



Podjela prostora (engl. *quad tree*, *area subdivision*)

Poligone raspodijelimo obzirom na relaciju s prozorom:

(1) poligon je izvan prozora (uklonimo ih iz liste)

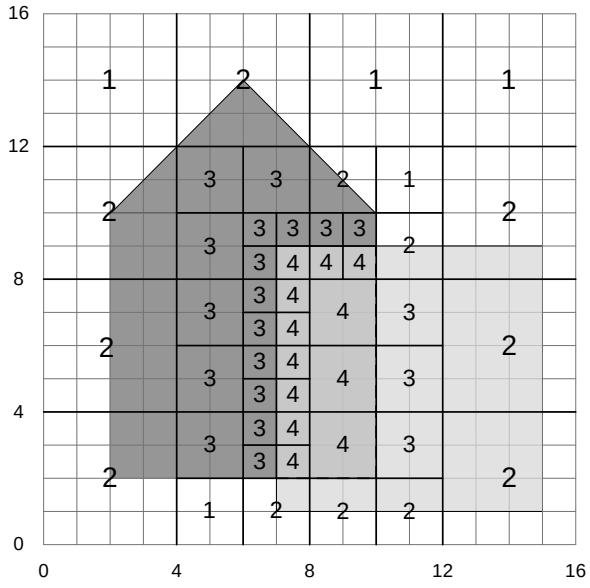
(2) poligon siječe prozor ili je u prozoru

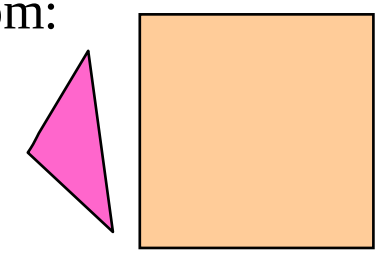
(3) poligon (jedan) prekriva prozor

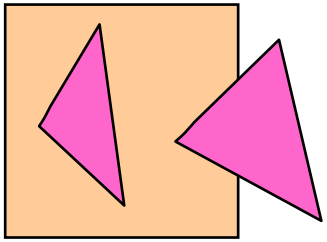
(4) više poligona prekriva ili siječe prozor

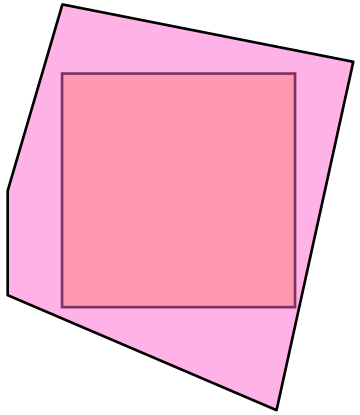
- daljnje ispitivanje – ako je jedan poligon koji prekriva prozor najbliži po udaljenosti od promatrača tada je (3)
- inače je slučaj (0)

- (minilekcije)



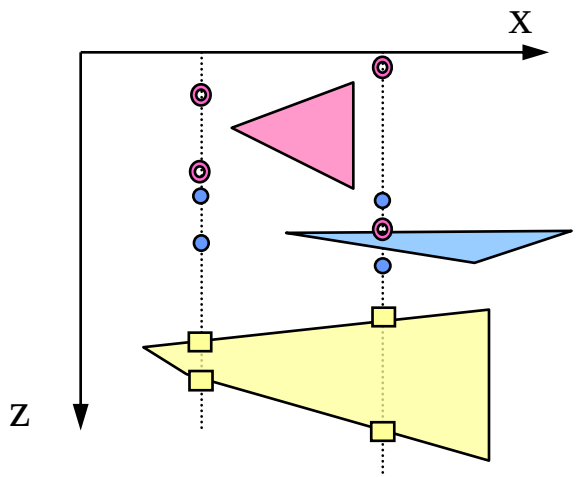




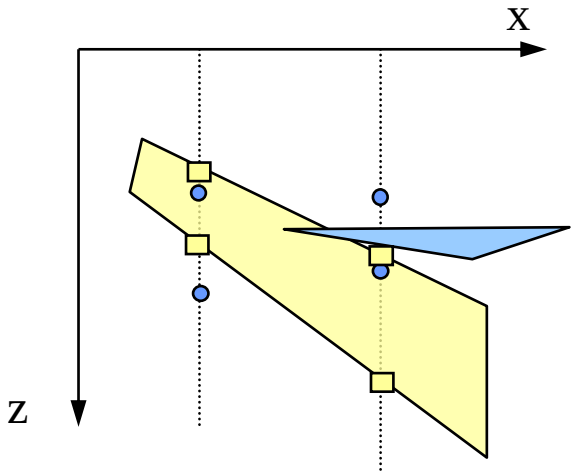


(4) daljnje ispitivanje – ako je jedan poligon koji prekriva prozor najbliži po udaljenosti od promatrača

- odredimo za svaki poligon ravninu u kojoj leži taj poligon i za tu ravninu z-koordinate u vrhovima promatranog prozora
  - ako je poligon koji prekriva prozor najbliži promatraču tada je to slučaj (4)
  - inače je scena u prozoru složena i dijelimo dalje, slučaj (0)



možemo odrediti poligon koji prekriva prozor i najbliži je



ne možemo odrediti poligon koji prekriva prozor i najbliži je

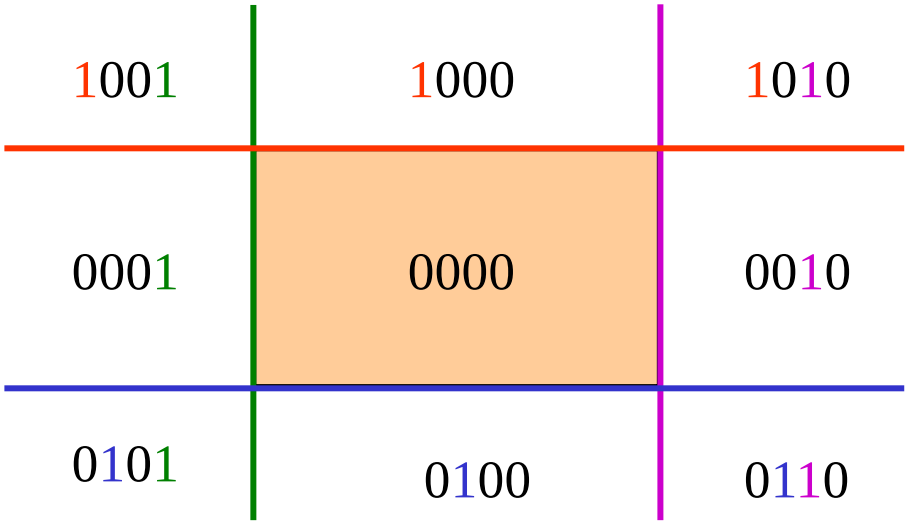
Ideja podjele prostora (četvero/oktalno stablo) koristi se kod brze pretrage scene izgradnja stabla 2D https://ciphrd.com/2020/04/02/building-a-quadtree-filter-in-glsl-using-a-probabilistic-approach/

3D https://collinhover.github.io/threeoctree/ Oktalno stablo http://potree.org/potree/examples/lion_las.html

BoundingBox npr http://ekelleyv.github.io/Flocking/ upotreba kod praćenje zrake npr. http://vcg.isti.cnr.it/spidergl/example.php?id=8

Odsijecanje dužina (engl. clipping) (prostor objekta, projekcije)

1. Algoritam Cohen-Sutherlanda
   - pridružimo pravce dužinama prozora
   - označimo područja sa četiri bita
     - bit 3 - 1xxx – područje iznad prozora – predznak od (y_{max}-y)
     - bit 2 - x1xx – područje ispod prozora – predznak od (y-y_{min})
     - bit 1 - xx1x – područje desno od prozora – predznak od (x_{max}-x)
     - bit 0 - xxx1 – područje lijevo od prozora – predznak od (x-x_{min})



– neka je pripadnost točke V_1 (V_2) području označeno četvero-bitnim kodom c_1 (c_2)

1. Dužina je trivijalno **vidljiva** ako je c_1=0000 i c_2=0000
2. Dužina trivijalno **nije vidljiva** ako je (c_1 AND c_2) ≠ 0000

   Npr. A=1001, B=1010, (A AND B) = 1000
3. Ako nije 1. ili 2. dužinu dijelimo pridruženim pravcima prozora. Segment koji je s vanjske strane odbacujemo. Postupak se ponavlja sve dok ne dobijemo trivijalan slučaj.

Neka je redoslijed provjere odozgo-prema-dolje (top-to-bottom) i s desna na lijevo.

Ovaj redoslijed odgovara redoslijedu bitova s lijeva na desno u kodu pridruženom točkama. 3210

http://min.nl/cs426/jar/clip.html

http://daign.github.io/clipping-with-caps/

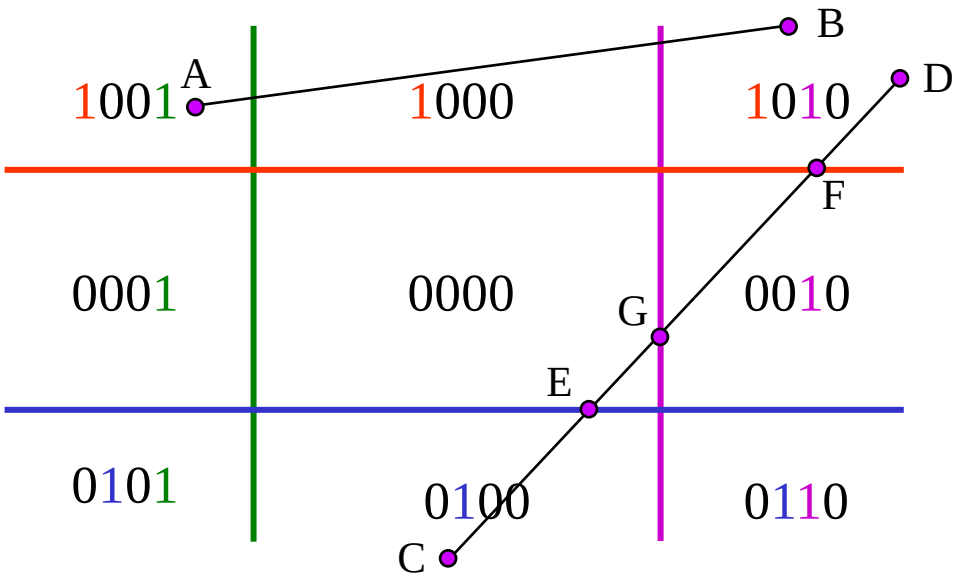
Npr.: C=0100 i D =1010 → E=0000 i D =1010 → E=0000 i F =0010 → E=0000 i G =0000

Sjecište E odredi se ($x= x_c+(x_d-x_c)(y_{min}-y_c)/(y_d-y_c)$, y_{min})

Ukoliko je u istom primjeru redoslijed točaka D =1010 i C =0100 → FC → GC → GE

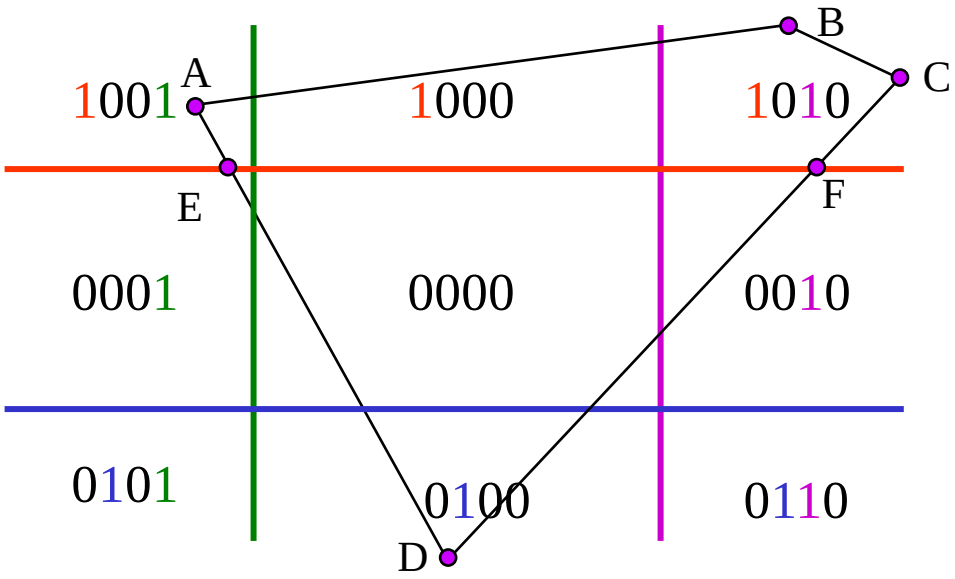
Postupak se jednostavno proširuje na više dimenzija.

Nedostatak – nepotrebno računanje "vanjskih" sjecišta npr. F.



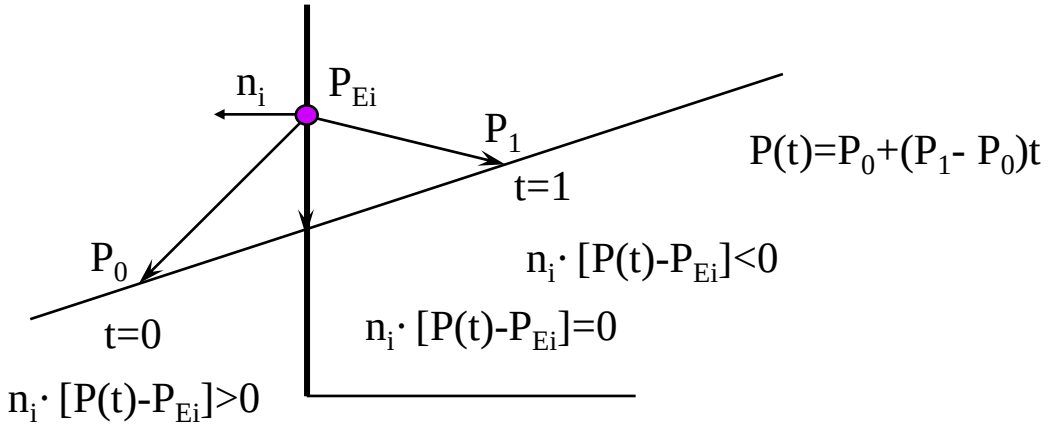
Odsijecanje poligona:

- načinimo odsijecanje svih linija poligona s y_{max}, i dobijemo poligon FDE,
- postupak nastavljamo s ostalim
- http://www.cs.rit.edu/~icss571/clipTrans/PolyClip.html



2. Algoritam Cyrus-Beck

- parametarski algoritam odsijecanja
- odsijecanje 2D dužine obzirom na konveksni poligon
- odsijecanje 3D dužine obzirom na konveksno tijelo



- Neka je E_i brid (ili poligon), a $\mathbf{n_i}$ normala na zadani brid (poligon).

Točka P_{EI} proizvoljna je točka na bridu (na primjer vrh).

Za sjecište pravca i brida vrijedi:

$$n_i \cdot [P(t)-P_{Ei}] = 0$$

Za P(t) uvrstimo jednadžbu pravca.

$$n_i \cdot [P_0+(P_1- P_0)\, t-P_{Ei}] = 0$$

$$n_i \cdot (P_0- P_{Ei})+ n_i \cdot (P_1- P_0)\, t = 0$$

$$t = \frac{n_i \cdot [P_0 - P_{Ei}]}{-n_i \cdot D}, \quad D = (P_1 - P_0)$$

Kako ne bi došlo do dijeljenja s nulom treba provjeriti

- $n_i \neq 0$, pogreška
- $D \neq 0$, mora biti $P_1 \neq P_0$
- $D \cdot n_i \neq 0$, $\overline{P_1P_0}$ je paralelna s bridom E_i

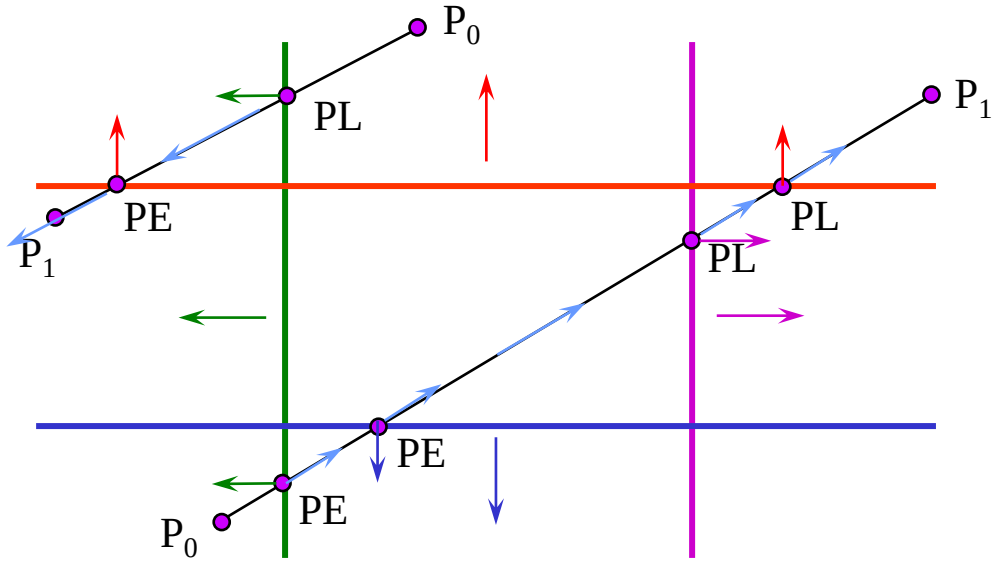
Presjecišta načinimo sa svim pravcima (poligona).

U slučaju prozora imamo najviše četiri sjecišta.

Parametar t mora biti $0 \leq t \leq 1$ pa inicijalno postavimo PE=0, PL=1;

Sjecišta podijelimo u skupove prema predznaku produkta $\overrightarrow{n_i} \cdot \overrightarrow{P_0P_1}$

- PE potencijalno ulazni - ako je predznak negativan, $\mathbf{kut}(\overrightarrow{\boldsymbol{n_i}}, \overrightarrow{\boldsymbol{P_0P_1}}) > \mathbf{90°}$
- PL potencijalno izlazni - ako je predznak pozitivan, $\mathbf{kut}(\overrightarrow{\boldsymbol{n_i}}, \overrightarrow{\boldsymbol{P_0P_1}}) < \mathbf{90°}$



U skupu PE odabere se onaj s najvećim parametrom (većim od nule), a u skupu PL onaj s najmanjim (ali tako da je manji od 1).

Ukoliko je $t_E > t_L$ dužina nije vidljiva. Koordinate sjecišta računamo sada kada imamo određene konačne parametre t_E, t_L. Nema suvišnih izračuna.

Algoritam vrijedi za konveksne poligone (poliedre).

Ako radimo s pravokutnim uspravnim prozorom možemo koristiti tablicu:

Brid odsijecanja	N_i	P_{EI}	P_0-P_{Ei}	$t = \frac{N_i \cdot [P_0 - P_{Ei}]}{-N_i \cdot D}$
LIJEVI $x = x_{min}$	(-1, 0)	(x_{min}, y)	(x_0-x_{min}, y_0-y)	$t = \frac{-(x_0 - x_{min})}{(x_1 - x_0)}$
DESNI $x = x_{max}$	(1, 0)	(x_{max}, y)	(x-x_{max}, y_0-y)	$t = \frac{(x_0 - x_{max})}{-(x_1 - x_0)}$
DONJI $y = y_{min}$	(0, -1)	(x, y_{min})	(x_0-x, y_0-y_{min})	$t = \frac{-(y_0 - y_{min})}{(y_1 - y_0)}$
GORNJI $y = y_{max}$	(0, 1)	(x, y_{max})	(x_0-x, y_0-y_{max})	$t = \frac{(y_0 - y_{max})}{-(y_1 - y_0)}$

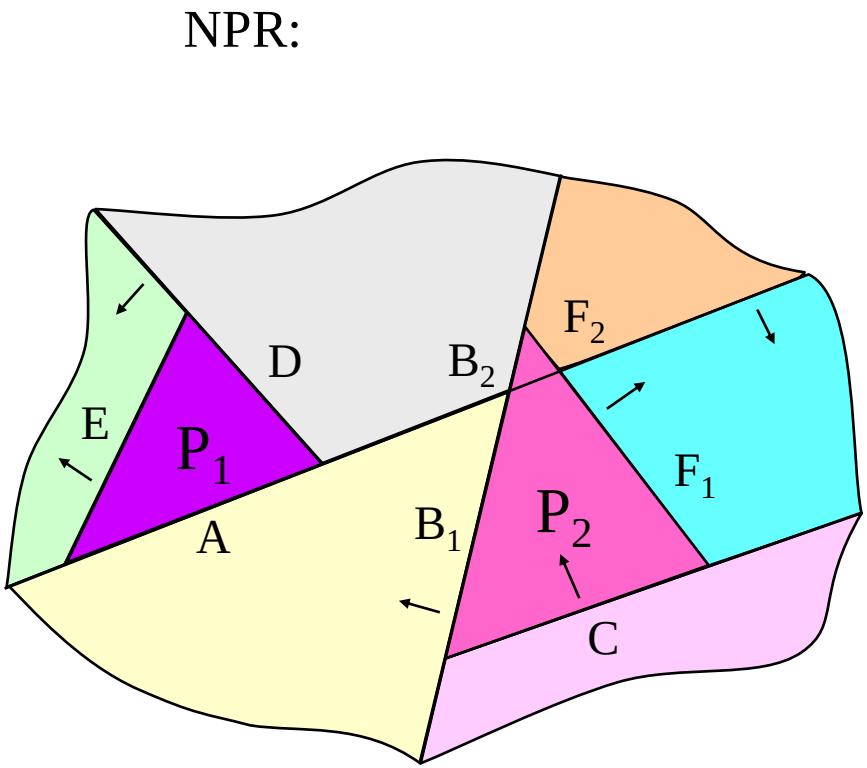
BSP – stabla (engl. Binary Space Partitioning)

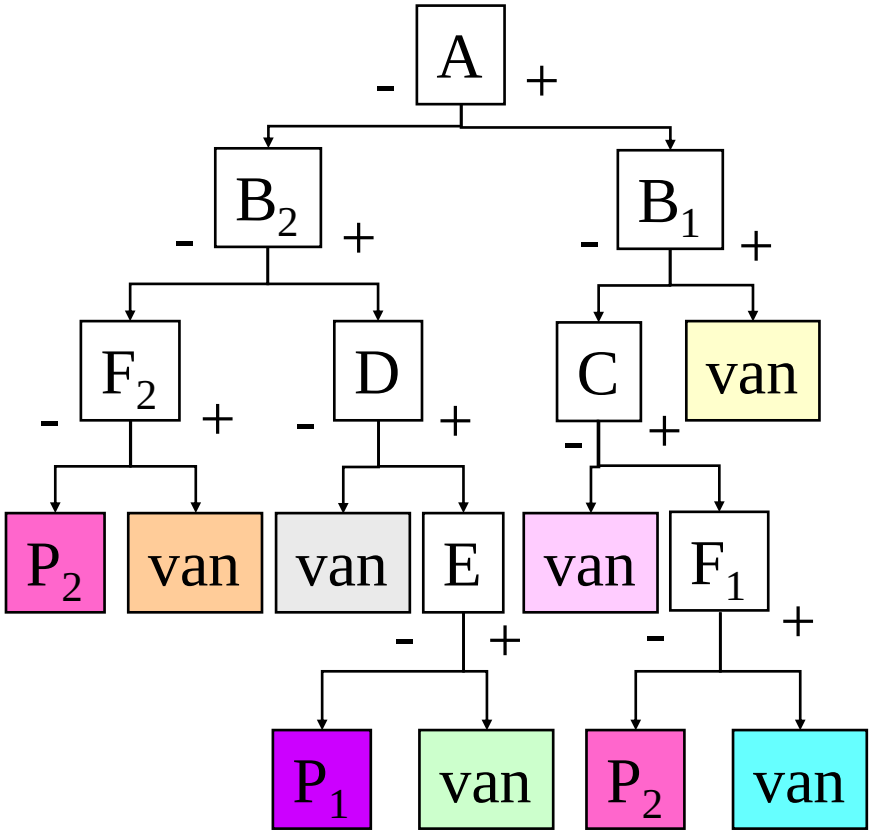
Binarna podjela prostora. Koristi se npr. za detekciju kolizija.

Algoritam je prikazan u dvije dimenzije i lako se proširi na tri dimenzije.

Prvo je potrebno sagraditi binarno stablo (balansirano).

NPR:





Pretraživanje stabla.

BSP – stabla

- pretraživanje stabla – množimo ispitnu točku s jednadžbama ravnina sve dok nije određen podprostor u kojem se točka nalazi
- određivanje vidljivosti

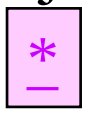
Po stablu iscrtavamo (tree->frontChild), (tree->root), (tree->backChild)

ovisno o položaju očišta po redoslijedu:

```
BSP_display(BSP_tree tree) {
    if (!EMPTY(tree)) {
        if (observer is located on front of root) {
            BSP_display(tree->backChild);
            displayPolygon(tree->root);
            BSP_display(tree->frontChild); }
        else {
            BSP_display(tree->frontChild);
            displayPolygon(tree->root);
            BSP_display(tree->backChild); } } }
```

(minilekcije) http://www.zemris.fer.hr/predmeti/irg/BSP/

Npr: BSP – određivanje vidljivosti

Neka je očište u području 

http://www.symbolcraft.com/graphics/bsp/index.php

https://halftheopposite.github.io/bsp-dungeon-generator/#/generate

Redoslijed iscrtavanja je:

E D B_2 F_2 A B_1 F_1 C

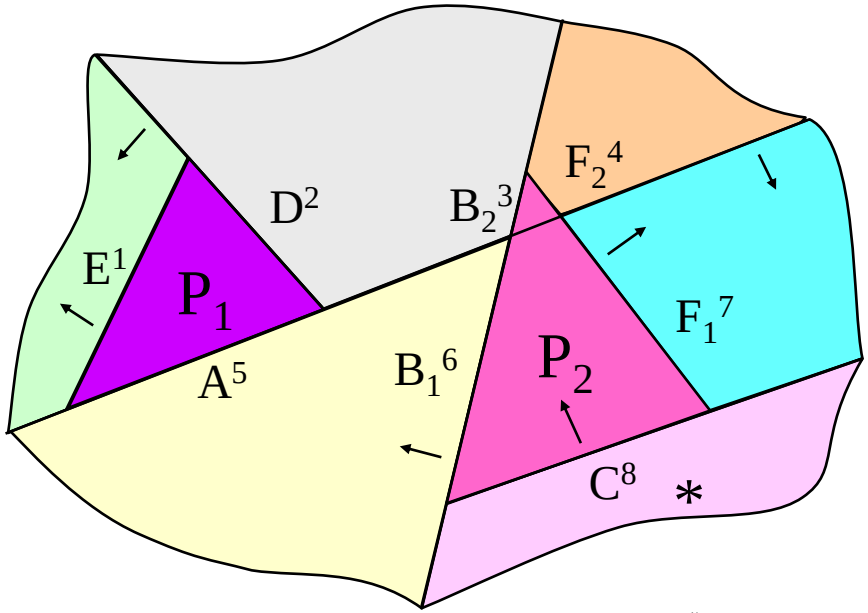
Iscrtavanje algoritmom Painter's.

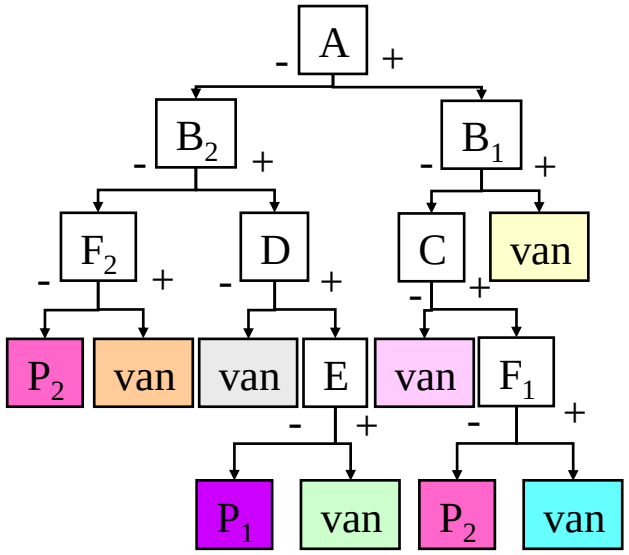
http://www.theparticle.com/applets/geometry/Q1FlightBox/index.html

http://www.theparticle.com/applets/wobble/index.html

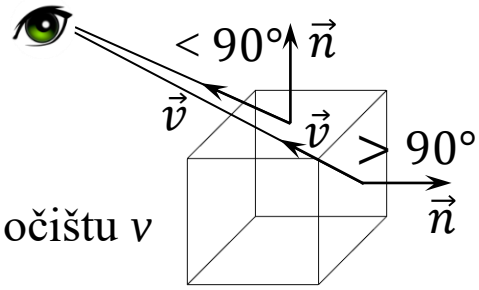
http://lab.passiomatic.com/quake3/

U dijelu izgradnje stabla samim algoritmom osigurano je da se netrivijalni slučajevi podjele u trivijalne segmente.





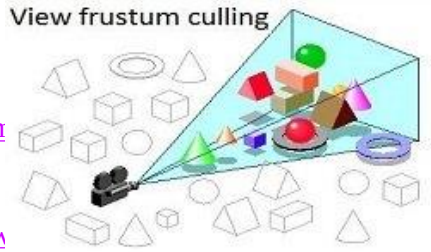
Uklanjanje poligona (engl. culling)

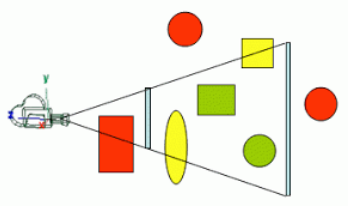


- uklanjanje stražnjih poligona
  Npr. na osnovi kuta između normale *n* i vektora prema očištu *v*
  ili orijentacije projiciranog poligona CW, CCW.

- uklanjanje poligona (objekata)
  izvan piramide pogleda

https://cedricpinson.github.io/osgjs-website/examples/frustun
https://schteppe.github.io/occlusion-culling.js/





- viška poligona https://threejs.org/examples/w
- uklanjanje zaklonjenih poligona (objekata) (engl. occlusion culling)

https://tsherif.github.io/webgl2examples/occlusion.html

Ukoliko objekti nisu prozirni možemo ukloniti stražnje poligone.

Ukoliko nema zrcaljenja možemo ukloniti poligone (objekte) koji nisu vidljivi.

Npr. u OpenGL:

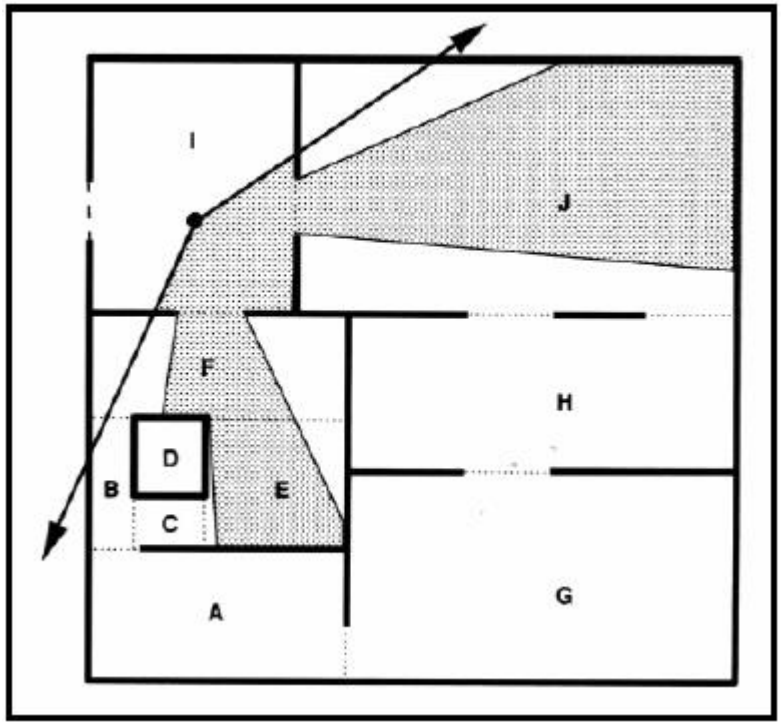
```
glFrontFace(GL_CW | GL_CCW);

glEnable(GL_CULL_FACE);
//glDisable(GL_CULL_FACE);
glCullFace(GL_FRONT | GL_BACK | GL_FRONT_AND_BACK);

glMaterialfv(GL_FRONT | GL_BACK | GL_FRONT_AND_BACK, …);
glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE | GL_FALSE);
```

Uklanjanje poligona (engl. culling) u zatvorenim prostorima

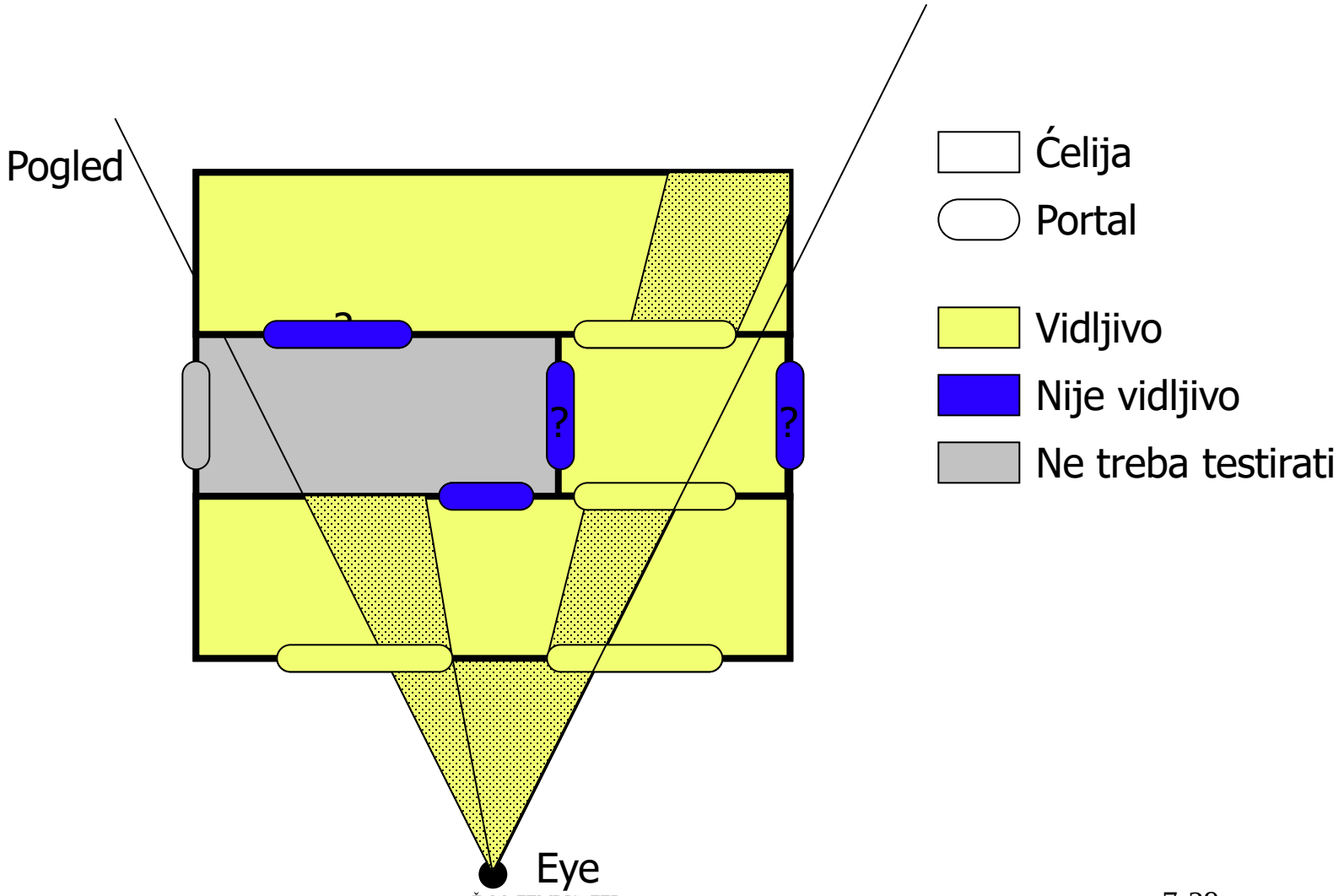
Potrebno je načiniti efikasne hijerarhijske grafove složenih scena.

- ćelije su prostorije ili dijelovi prostorija - A, B, C, D, …
- portali su vrata između pojedinih prostorija

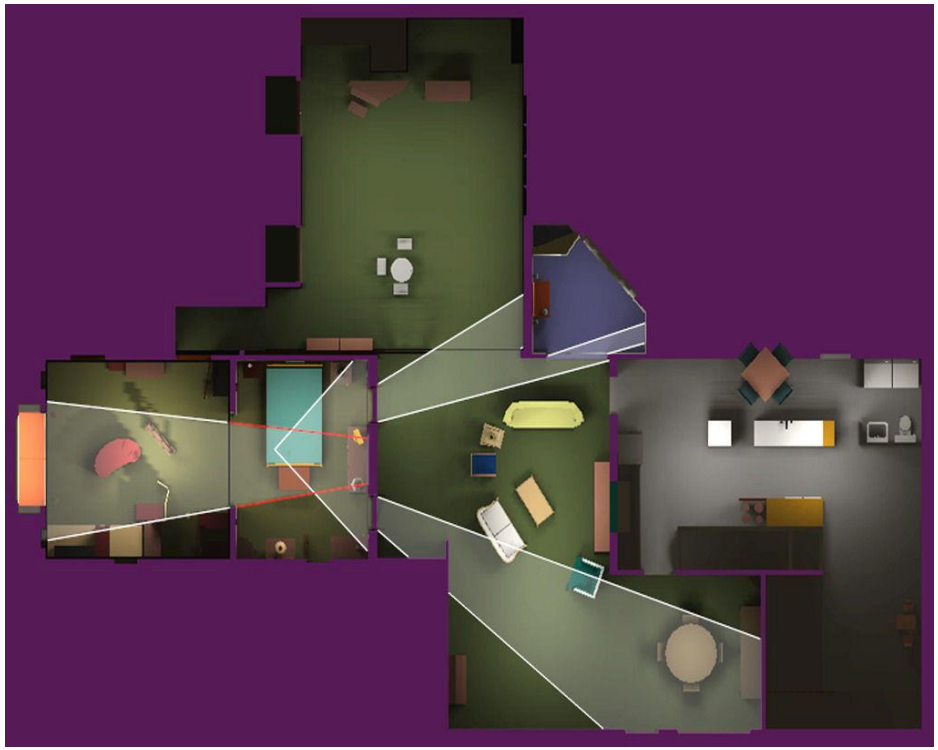
Ovisno o položaju promatrača i smjeru pogleda otvaraju se pojedini portali i ćelije.



Ćelije i portali

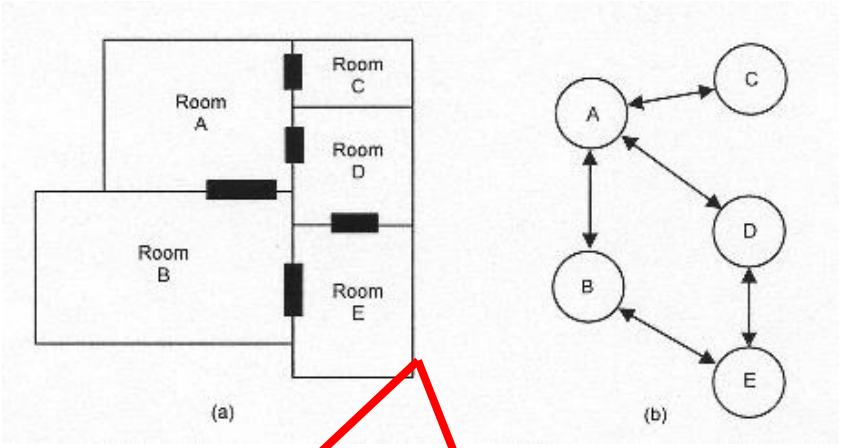


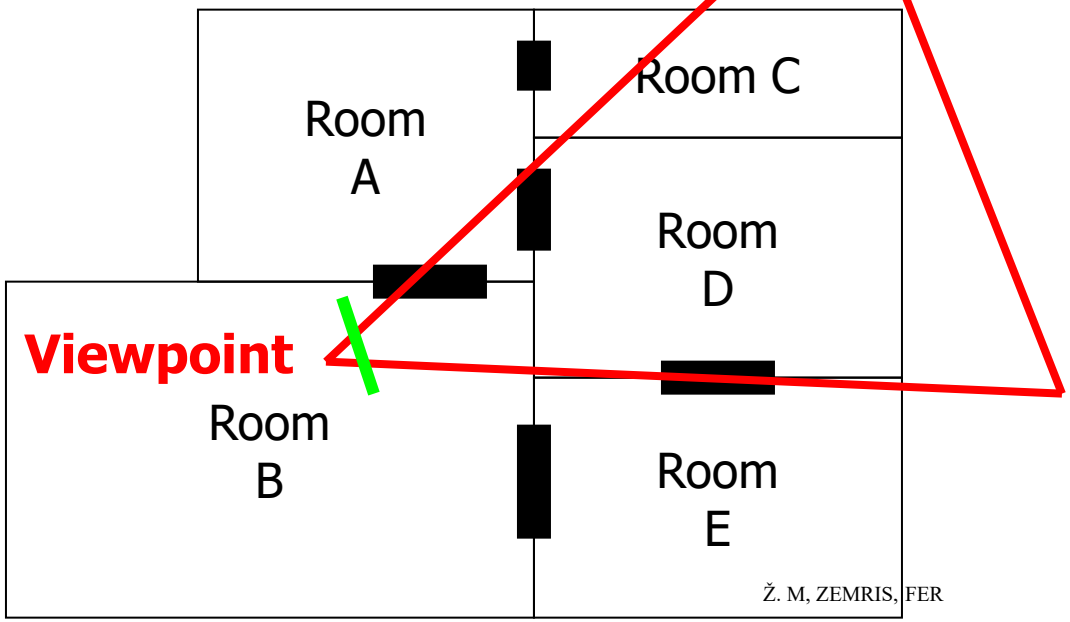
- primjer ćelija i portala
  - pogodni za unutarnje prostore
  - efikasni (brzo rade i malo memorije zauzimaju)



PVS potencijalno vidljivi skup (engl. Potentially visible set)

http://media.tojicode.com/q3bsp/





PVS = B, A, D

Kod otvorenih scena (tereni) - hijerarhijski grafovi - promjena složenosti prikaza (engl. LOD level of detail)

Veliki broj udaljenih i skrivenih poligona https://www.mvrsimulation.com/casestudies/f16-luke.html

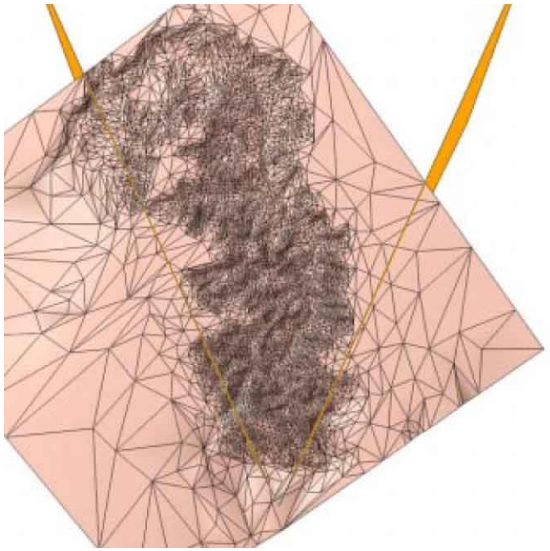
Unaprijed se pohrane objekti s različitim brojem poligona (LOD).

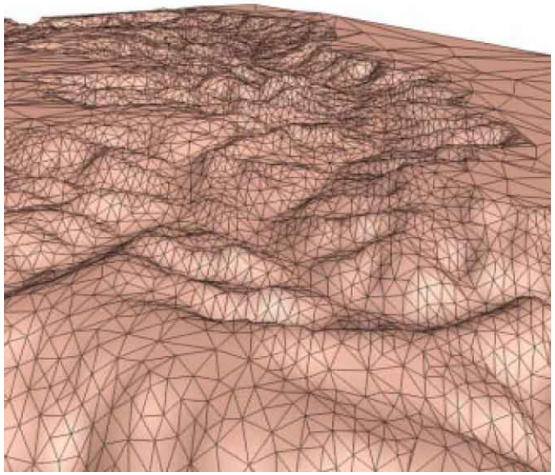
Ovisno o udaljenosti prikazuje se različita razina složenosti a između se interpolira.

Kako objekti ne bi „iskakali" koristi se postupno pojavljivanje u prikazu ili geomorfi

Kontinuirani geometrijski preobražaj jednog oblika u drugi (*morphing*), geomorfi.

https://regl-project.github.io/regl/www/gallery/geomorph.js.html http://www.zephyrosanemos.com/windstorm/current/live-demo.html





Grafički protočni sustav – uklanjanje skrivenih linija i površina

- cilj je u grafičkom protočnom sustavu ukloniti što prije dijelove scene koji nisu vidljivi pa koristimo različite strukture podataka (oktalno, četvero stablo, BSP, omeđujući volumeni)
- moramo paziti ako imamo zrcalne objekte (refleksije), prozirne objekte, sjene da ne uklonimo objekte prerano
- za zatvorene prostore možemo koristiti npr. ćelije, portali i PVS za otvorene prostore - LOD
- prvo koristimo jednostavnije i računski jeftinije postupke,
- odsijecanje poligona van piramide pogleda (engl. view volume clipping) (Cohen-Sutherland, Cyrus-Beck)
- uobičajeno koristimo uklanjanje stražnjih poligona (engl. back face culling)

- rasterski sustav (prostor slike)
  - Painter's algoritam uz BSP sortiranje, Watkins, Warnock, Z-spremnik logaritamski http://threejs.org/examples/#webgl_camera_logarithmicdepthbuffer, perspektivno ispravna interpolacija z-koordinate

Moguće uštede ako radimo u 2D, 2,5 D - stapanje u prostoru projekcije

– ako je pozadina scene statična, u poseban sloj (engl. layer) crta se statični dio
  - objekti koji su pokretni crtaju se u poseban sloj (može biti i mali pravokutnik)
  - za pojedini sloj moguće je koristiti i spremnik maske (kombiniranje slojeva)

U svim algoritmima prisutni su problemi zaokruživanja i numeričkih pogrešaka koji izazivaju vidljive učinke na rezultatu.

Pogreška u Z-spremniku (premala dubina z-spemnika ili raspon z-koordinate)

Z-fighting:

http://www.realtimerendering.com/erich/udacity/exercises/unit2_zfighting_exercise.html

http://www.realtimerendering.com/erich/udacity/exercises/unit2_zfighting_solution.html