Látható felszín algoritmusok

Látható felszínek

Z-buffer algoritmus

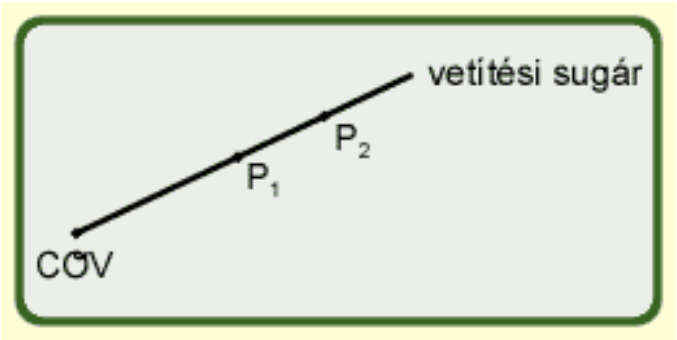
Festő algoritmus

A látható felszín meghatározására szolgáló algoritmusok

A tárgyak takarják-e egymást?
Mely tárgy látható?

Pontokra: $P_1 = (x_1, y_1, z_1)$ és $P_2 = (x_2, y_2, z_2)$
Takarja-e egyik a másikat?

Ha ugyanazon a vetítési sugáron vannak, akkor a közelebbi takarja a másikat, különben nem.



Nehéz probléma.

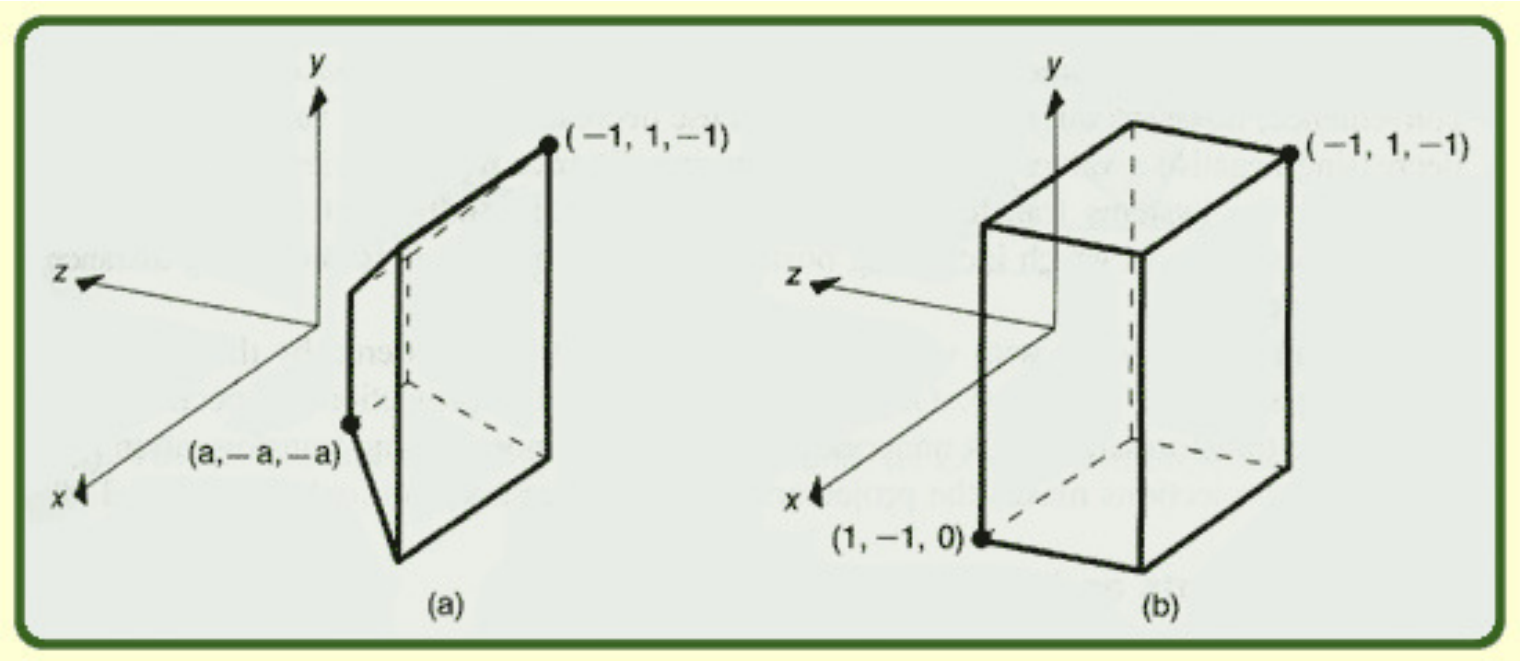
Mélységbeli összehasonlítás

Helye: a normalizálási transzformáció után, ekkor

1. **párhuzamos vetítésnél**: a vetítő sugarak párhuzamosak a z-tengellyel, ekkor P_1 és P_2 ugyanazon a vetítési sugáron van, ha
$$x_1 = x_2 \text{ és } y_1 = y_2$$
2. **perspektív vetítésnél**: a vetítési sugarak a COV-ből indulnak ki, ekkor P_1 és P_2 ugyanazon a vetítési sugáron van, ha
$$\frac{x_1}{z_1} = \frac{x_2}{z_2} \text{ és } \frac{y_1}{z_1} = \frac{y_2}{z_2}$$

A látható felszín meghatározására szolgáló algoritmusok

Perspektív vetítésnél azt a transzformációt használjuk, amely a perspektív kanonikus térfogatot átviszi párhuzamos kanonikus térfogatba.



perspektív kanonikus térfogat

párhuzamos kanonikus térfogat

A látható felszín meghatározására szolgáló általános algoritmusok

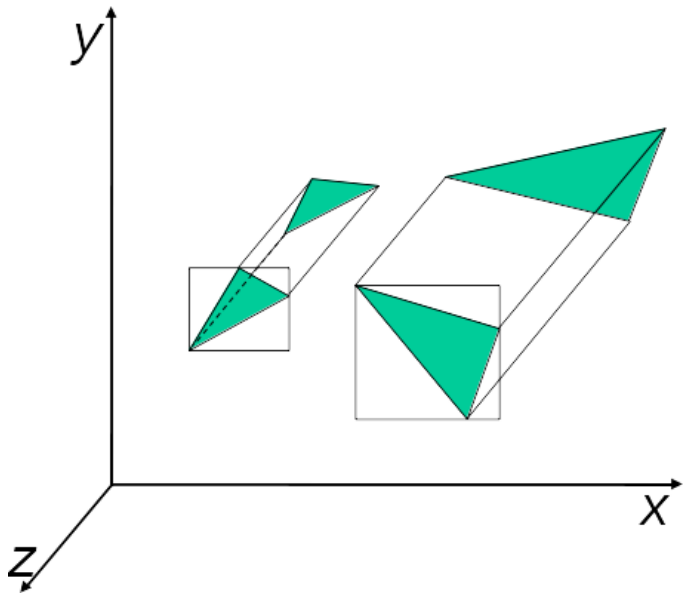
Perspektív kanonikus térfogatot párhuzamos kanonikus térfogatba transzformáló mátrix:

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{1+z_{min}} & \frac{-z}{1+z_{min}} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Ekkora vetítési sugarak már párhuzamosak a z-tengellyel. Egyszerűbben végezhető a vágás.

A látható felszín meghatározására szolgáló általános algoritmusok

Tárgyak kiterjedése, határoló téglalapok, testek

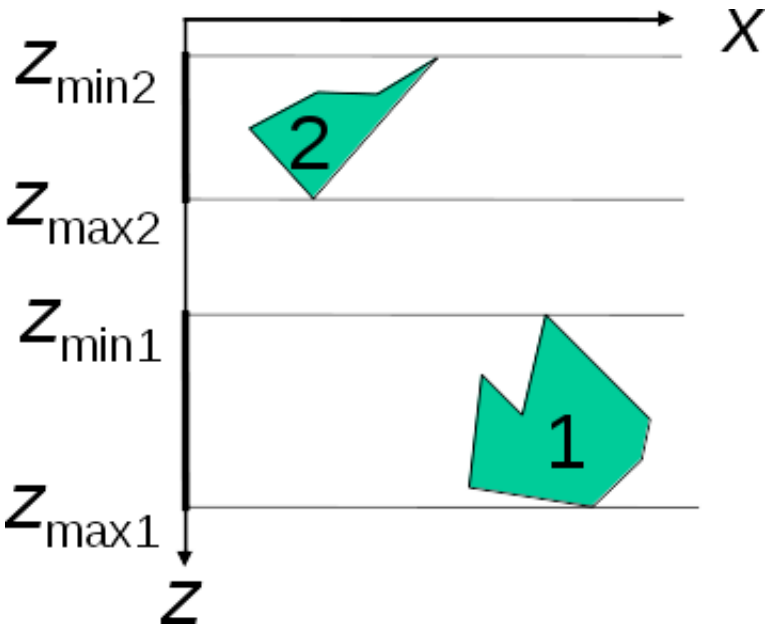


Határoló téglalap teszt
Ha a határoló téglalapok nem fedik egymást, akkor a vetületek sem fedik egymást, különben további vizsgálat szükséges.

A látható felszín meghatározására szolgáló általános algoritmusok

1-dimenziós kieterjedés (határoló intervallum)

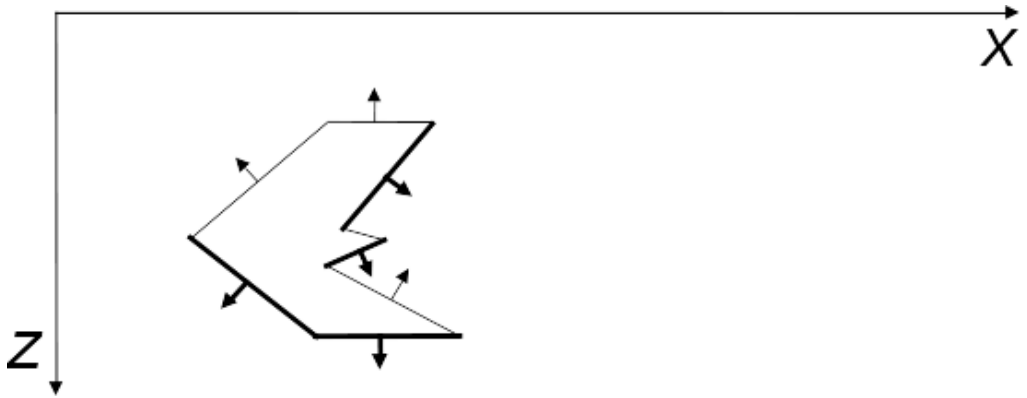
minmax-teszt
A kiterjedés minimális és maximális értékeinek összehasonlításával döntjük el a takarást.



A kiterjedés meghatározása: a tárgy (csúcs)pontjaihoz tartozó koordináták *min.* és *max.* értékeiből.

Hátranéző lapok kiválogatása

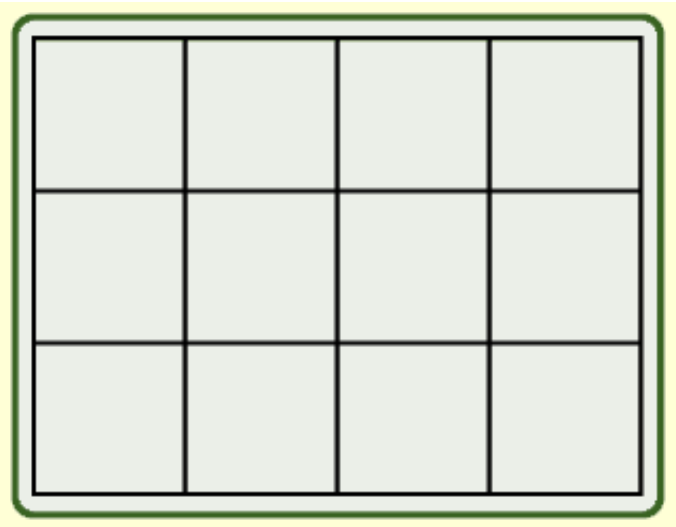
Tegyük fel, hogy poligon határú síklapokkal határolt a tárgy, és adottak a síklapoknak a tárgyból kifelé mutató normálisai. Ekkor azok a lapok nem láthatók, amelyek normálisai a „megfigyelőtől ellentétes” irányba mutatnak.



A látható felszín meghatározására szolgáló általános algoritmusok

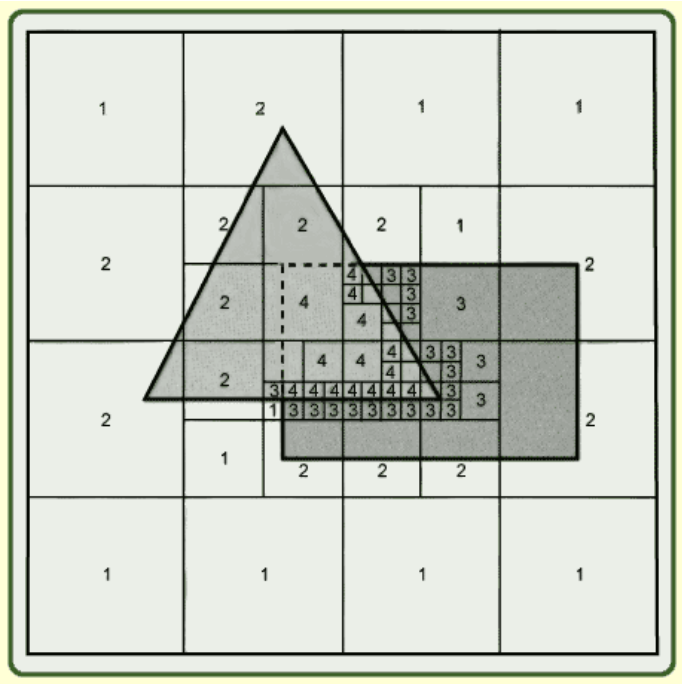
Térbeli partícionálás

Észrevétel: nem minden tárgynak van minden vetítési sugárral metszéspontja (pl. távol vannak, más irány) $\Longrightarrow$ osszuk fel (particionáljuk) a képernyőt.



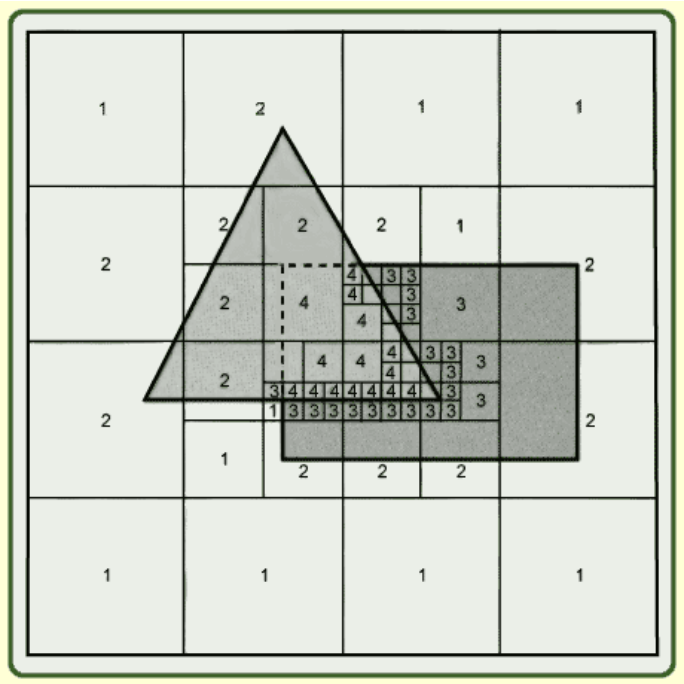
Meghatározzuk, hogy mely tárgyak vetülete van benne a megfelelő részben (partícióban) és csak azokkal keresünk metszéspontokat.

A látható felszín meghatározására szolgáló általános algoritmusok



Ez jó módszer, ha a tárgyak vetületei egyenletesen oszlanak el a teljes képernyőn, különben különböző méretű partíciókat érdemes készíteni, kisebb partíciót ott, ahol több tárgy vetülete van.

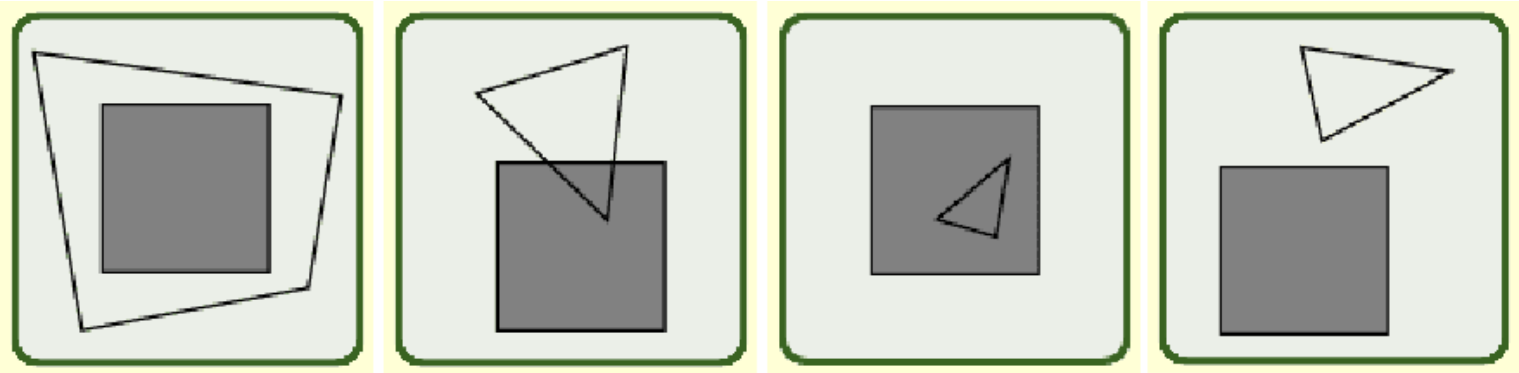
Területosztó algoritmus látható felszín meghatározására: „oszd meg és uralkodj”



Ha egy területen könnyen eldönthető, hogy melyik poligon jeleníthető meg, akkor azt rajzoljuk ki, különben osszuk fel a területet és alkalmazzuk az eljárást a részterületekre.

Látható felszín algoritmusok

Négy lehetőség egy poligon és egy téglalap alakú terület átfedése között:



Tartalmazó poligon

Metsző poligon

Tartalmazott poligon

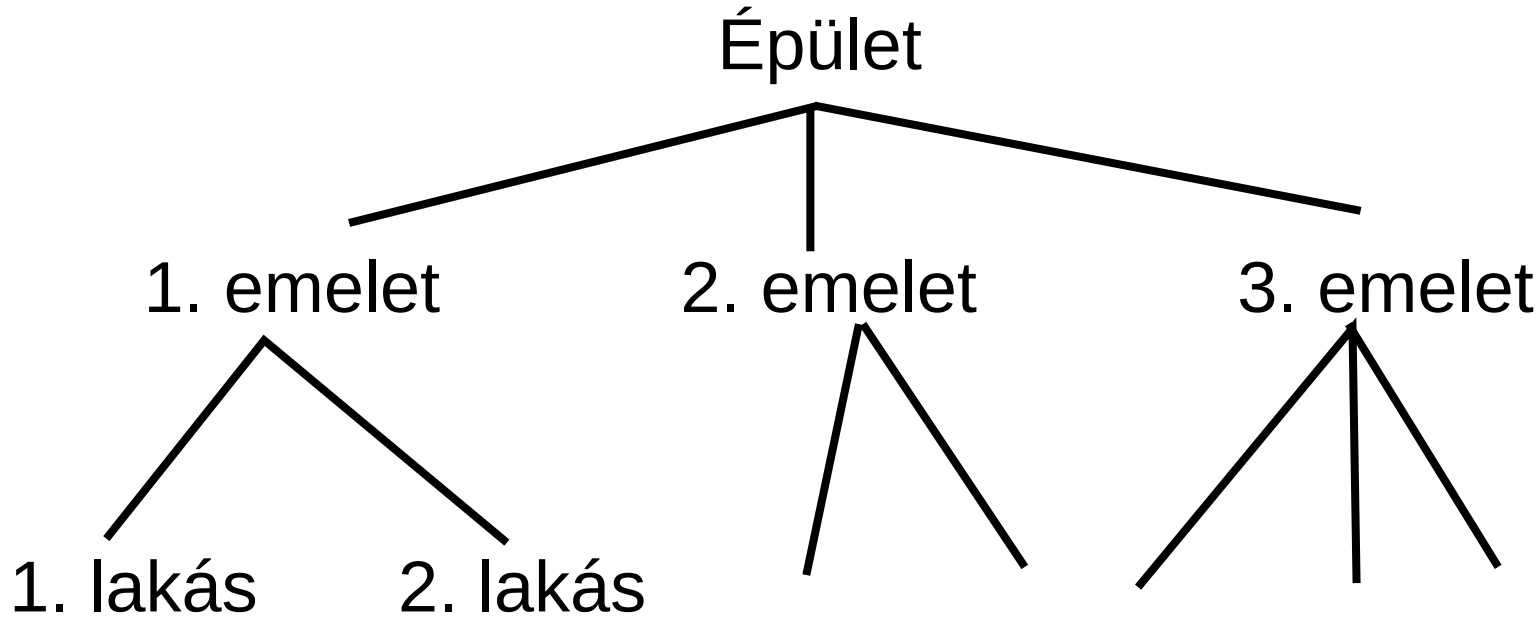
Idegen poligon

Látható felszín algoritmusok

Mikor dönthető el könnyen, hogy mi rajzolható?

1. Minden poligon idegen a területtől (háttér)
2. Egyetlen metsző vagy tartalmazott poligon (háttér + pásztázással poligon)
3. Egyetlen tartalamazó poligon (rajzolás a poligon színével)
4. Van olyan tartalmazó poligon, amelyik a többi előtt van.

Hierarchikus struktúrák alkalmazása



Ha a vetítési sugár nem metszi az épületet, akkor az emeleteit és az emeletek lakásait sem metszi (tehát nem kell vizsgálni azokat).

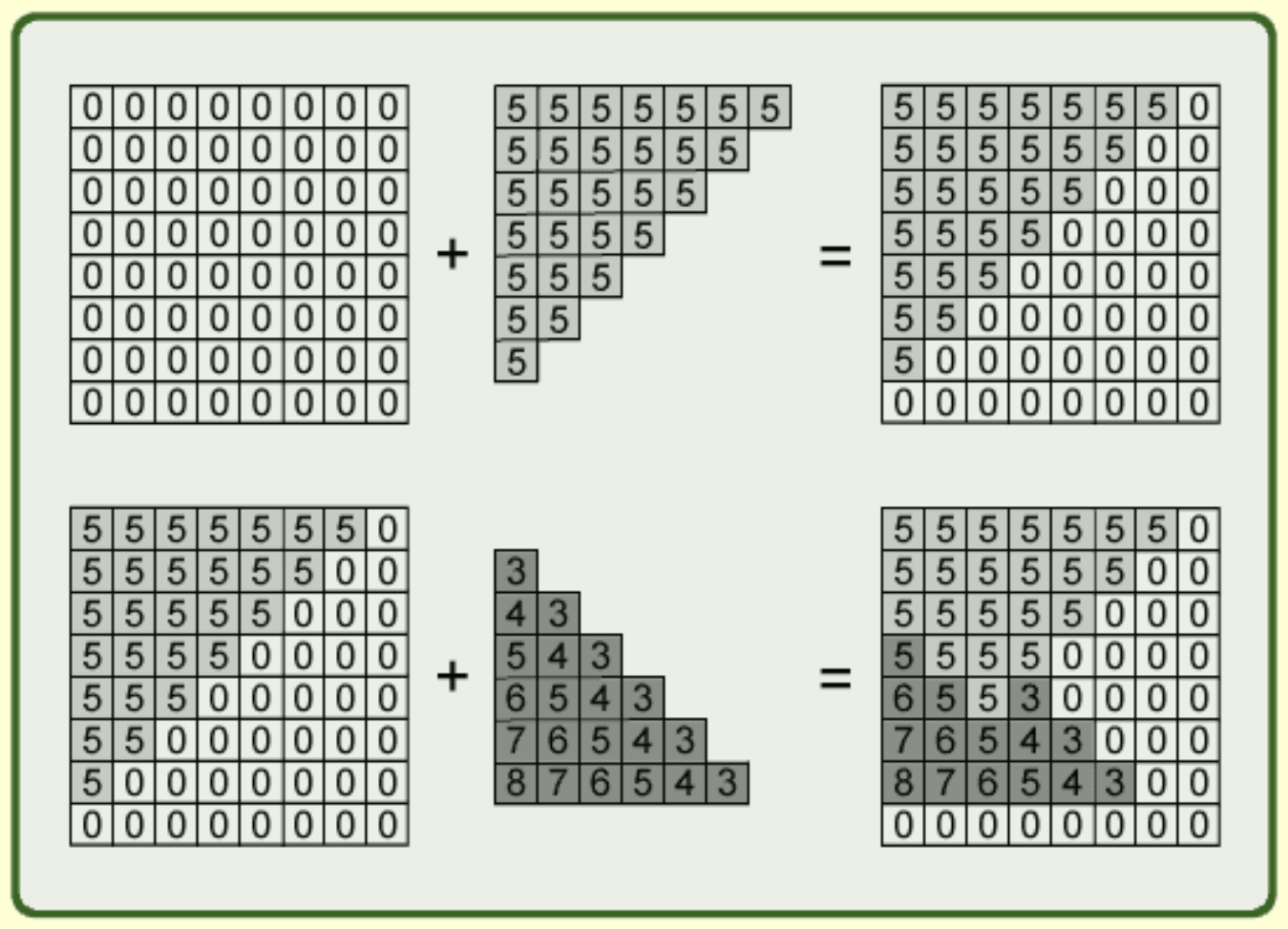
Z-buffer algoritmus (kép alapú)

F: kép-puffer (képpontok tárolására) kezdeti értéke a háttérszín

Z: mélység-puffer (minden pontban a megfelelő z érték), kezdeti értéke a z-max (hátulsó vágási sík)

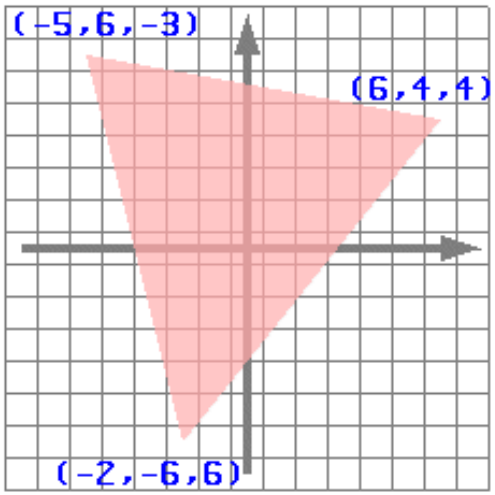
Pásztázás közben F-be és Z-be bekerül az új pont, ha nincs messzebb, mint az eddigi z érték.

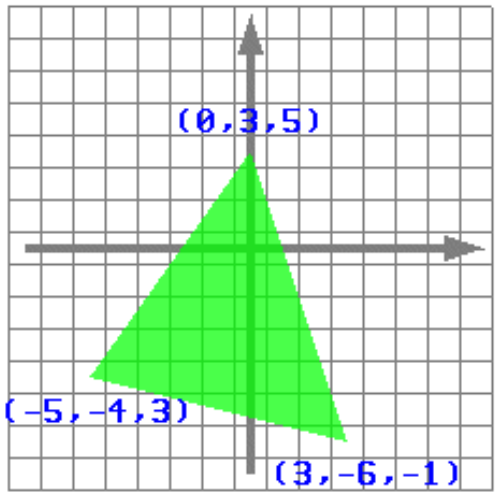
Z-buffer algoritmus

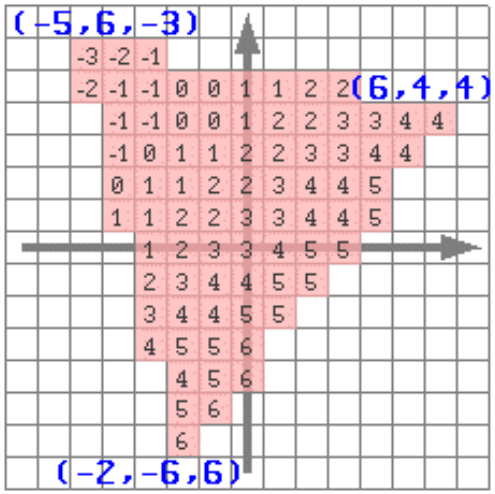


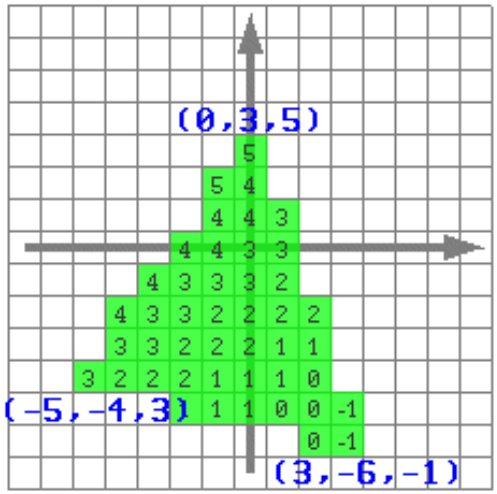
A hátsó vágási sík a $z = 0$

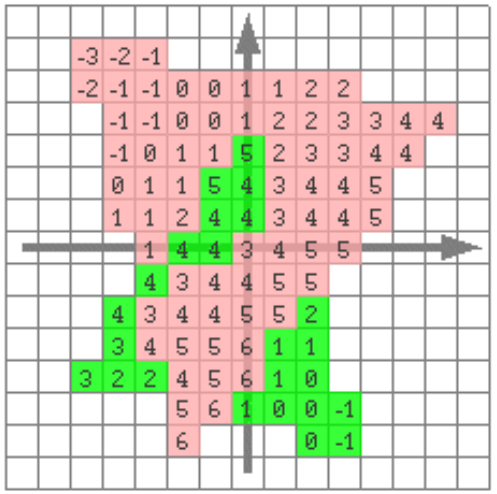
Z-buffer algoritmus



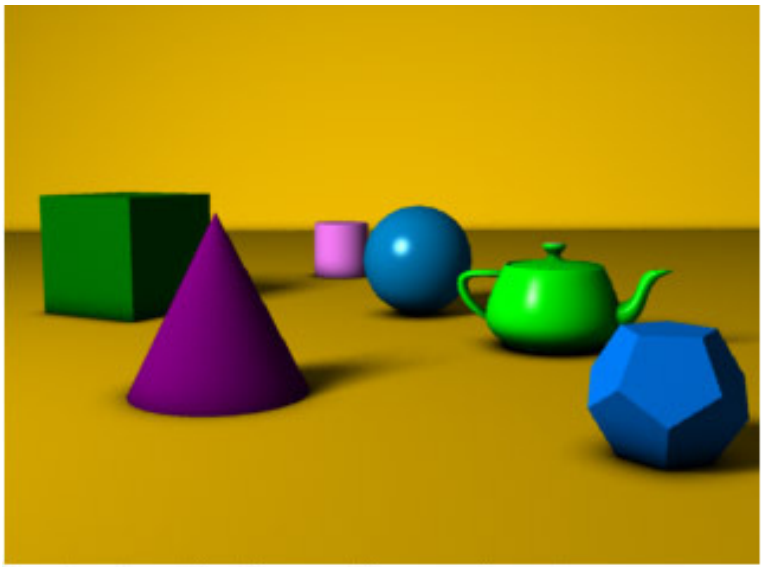








Z-buffer algoritmus



Kép

Z-buffer

Z-buffer algoritmus

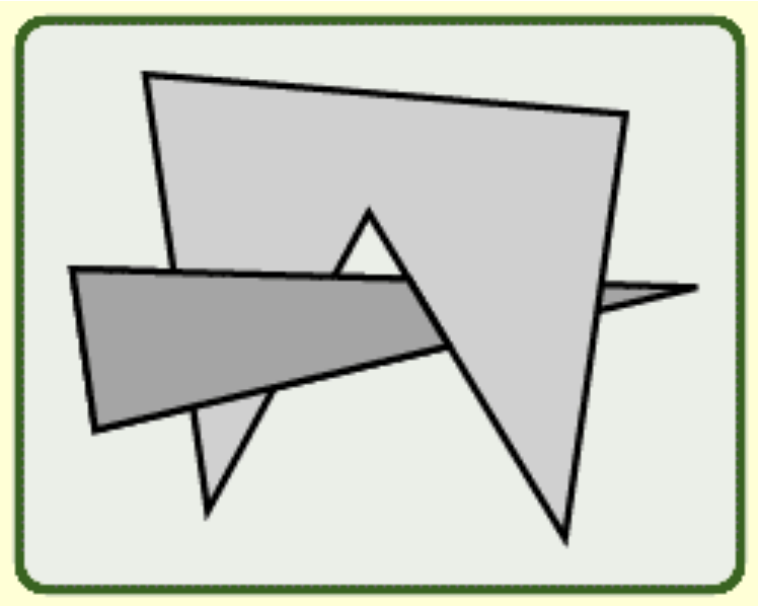
Tulajdonságai:

- Nincs tárgyak rendezése, összehasonlítása, metszéspontok számítása.
- Poligononként végezhető el, ha nincsenek átható poligonok.
- Nem csak poligonokra jó.
- Nagy helyigén, de lehet sávonként haladni.
- Könnyű implementálni.
- Könnyű egy újabb tárgy képét hozzávenni.

Lista prioritásos algoritmusok

Meghatározzák a tárgyaknak azt a sorrendjét, ami a kép kirajzolásához kell.

Például, ha a z értékekben nincs átfedés, akkor a tárgyakat növekvő z értékük szerint kell rendezni, és utána távolról közelre haladva megjeleníteni.
Néha még akkor is lehet ilyen sorrendet megadni, ha a z értékekben van átfedés, de nem mindig.



Ilyenkor szétvágjuk a tárgyakat és a darabokat rendezzük sorba.

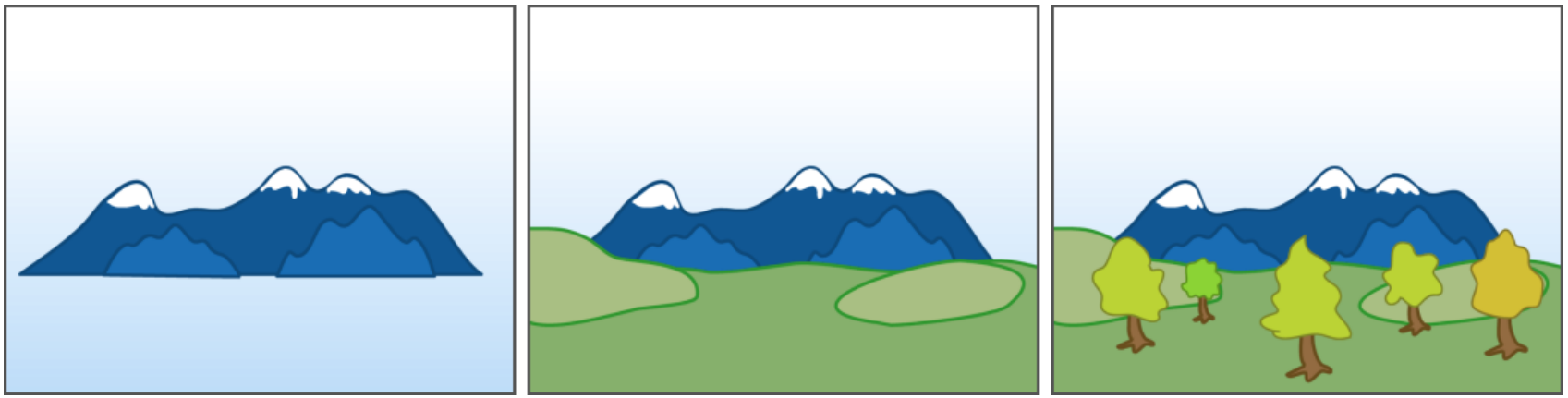
Mélység szerint rendező algoritmus

Lépések:

1. Rendezzük a poligonokat legtávolabbi z koordinátájuk szerint.
2. Vágjuk szét az átfedő poligonokat (ha szükséges)
3. Pásztázzunk minden poligont hátrulról előre haladva.

Ha a poligonok párhuzamos síkokban fekszenek, akkor a 2. lépés kimaradhat.

Festő algoritmus



Takarás vizsgálatok

Tegyük fel, hogy a P poligon legtávolabbi z koordinátája szerint a lista végén van.
Pásztázás előtt össze kell hasonlítani a lista azon Q elemeivel, amelyeknek a z irányú kiterjedése átfedi P z irányú kiterjedését, és meg kell vizsgálni, hogy

P eltakarja-e Q-t?

Takarás vizsgálatok

P eltakarja-e Q-t?

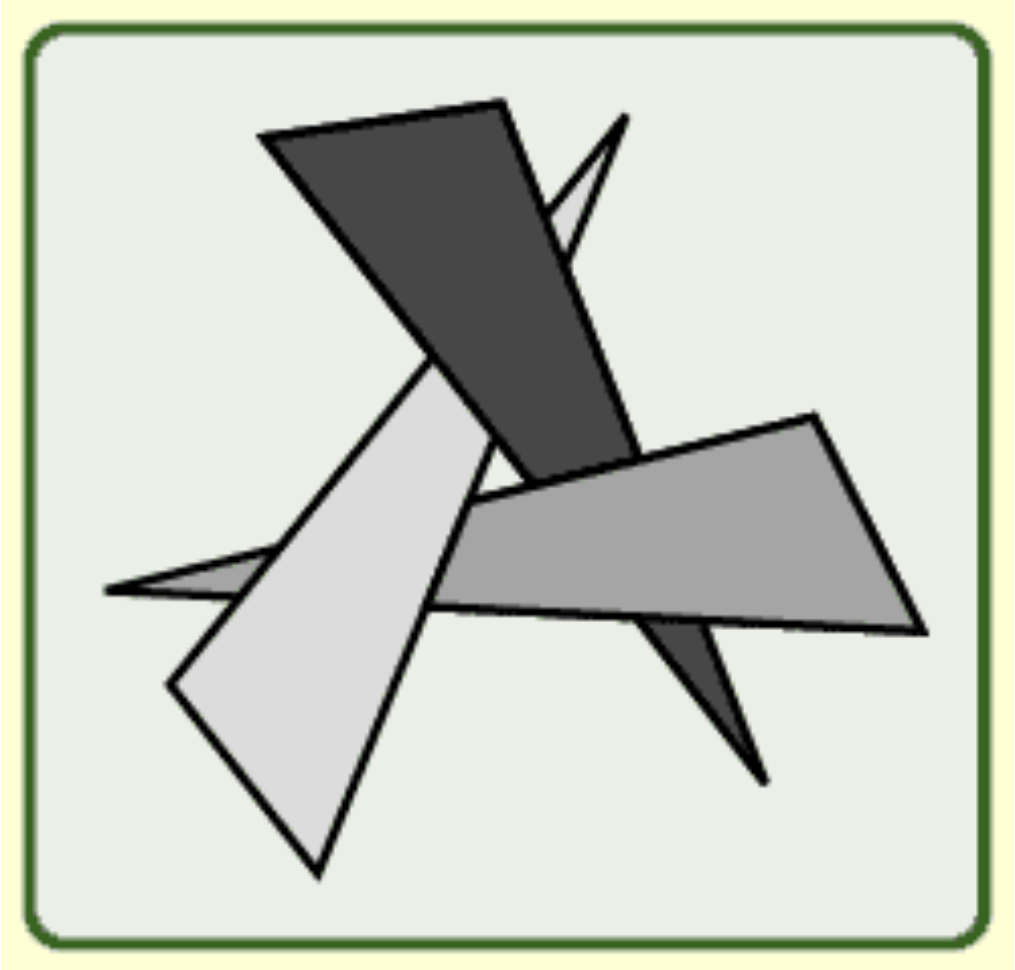
1. Ha P és Q x kiterjedése nem átfedő, akkor **nem**.
2. Ha P és Q y kiterjedése nem átfedő, akkor **nem**.
3. Ha COV-ból nézve P teljes terjedelmében a Q skíjának túlsó oldalán van, akkor **nem**.
4. Ha COV-ból nézve Q teljes terjedelmében P síkjának az innenső oldalán van, akkor **nem**.
5. Ha P és Q (x, y) síkra való vetülete nem átfedő, akkor **nem**

3' Ha COV-ból nézve Q teljes terjedelmében a P sík túlsó oldalán van, akkor P Q csere.

4' Ha a COV-ból nézve P teljes terjedelmében Q-nak az innenső oldalán van, akkor P Q csere.

Különben P-t vagy Q-t fel kell darabolni a másik síkkal és a darabokat beilleszteni a listába.

Takarás vizsgálatok



Végtelen ciklust eredményezne, ezért megjelöljük azokat a poligonokat, amelyeket egyszer már a lista végére tettünk, és ha újra előjönnek, akkor darabolunk.

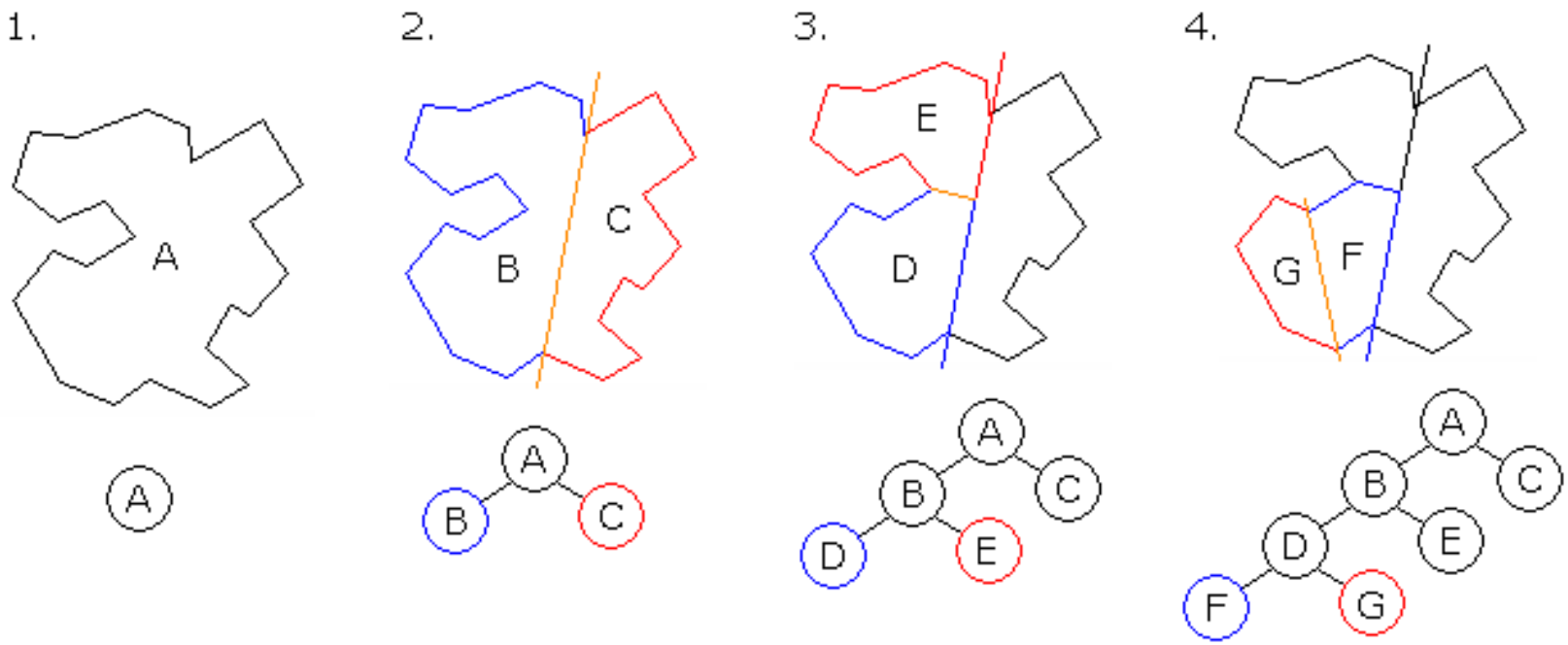
Bináris tér-partícionáló fa algoritmusok

Ötlet Ha van olyan sík, amely a tárgyakat (teljes egészükben) két féltérbe osztja, akkor a COV-t tartamazó féltér tárgyait nem takarjá el a másik féltér tárgyai

BSP fa Csomópontok - poligonok
A csomóponthoz tartozó poligon síkjával darabolhatjuk a többi poligont, és azok darabjaival folytathatjuk a fát

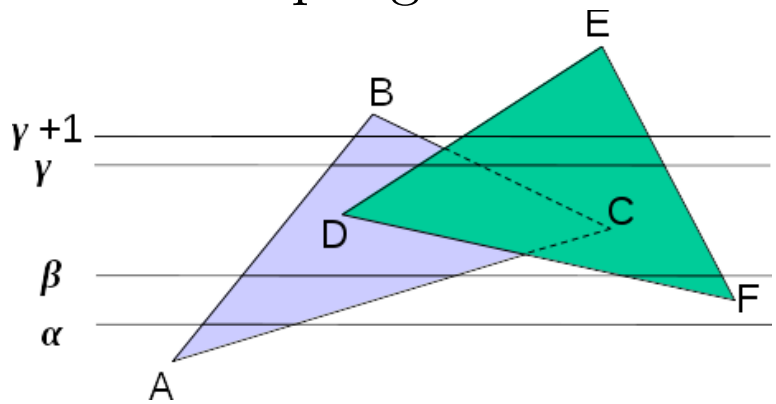
- **bal** oldalra: azok a poligonok, amelyek elöl vannak (később kell rajzolni)
- **jobb** oldalra: azok a poligonok, amelyek hátul vannak (korábban kell rajzolni)

Bináris tér-particionálás



Pásztázó vonal algoritmus

Hasonló a poligonok kitöltését végző algoritmushoz.



Vízszintesen pásztázunk

Most több poigon lehet.

Pásztázó vonal algoritmus

ÉT (élek táblázata, lásd poligon kitöltése): A kisebbik y értékük alapján szerint rendezve az összes élet tartalmazza. A vízszintes élek kimaradnak!

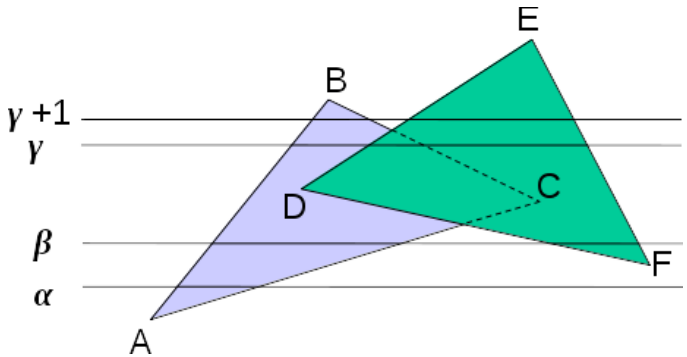
Annyi lista van, ahány pásztázó vonal. Minden listában azok az élek szerepelnek, amelyek alsó végpontja a pásztázó vonalon van. A listák az élek alsó végpontjának x koordinátája, ezen belül a meredekség reciproka szerint rendezettek.

Minden lista elem tartalmazza az él y_{max} és x_{min} koordinátáját és a meredekség reciprokát és a **poligon azonosítóját** (több poligon lehet).

Pásztázó vonal algoritmus

PT (poligonok táblázata):
egy elem részei:

- azonosító
- a sík egyenletének együtthatói
  a sík egyenlete: $Ax + By + Cz + D = 0$
- árnyalati / színezési információ
- `ki`/`be` jelző (kezdő érték `ki`)



ÉT:

AB	AC	
FD	FE	
CB		
DE		

PT:

ABC	
DEF	

Pásztázó vonal algoritmus

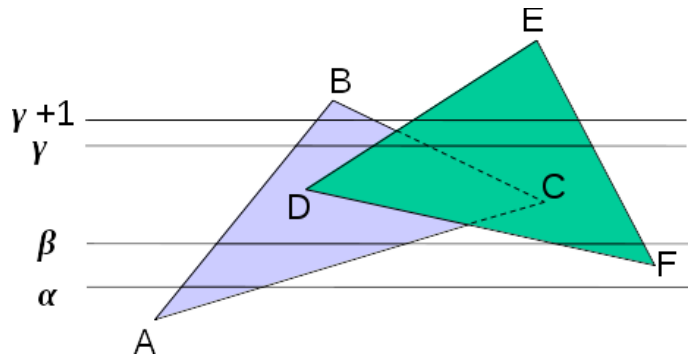
AÉT (aktív élek táblázata):
Alulról felfelé, balról jobbra haladva

α: $\underbrace{AB, AC}_{ABC}$ AB-nél `be`-, AC-nél `ki`kapcsolva

AB-től AC-ig ABC színe

β: $\underbrace{AB, AC}_{ABC}$, $\underbrace{FD, FE}_{DEF}$

`be` `ki` `be` `ki`



Pásztázó vonal algoritmus

AÉT (aktív élek táblázata):
Alulról felfelé, balról jobbra haladva

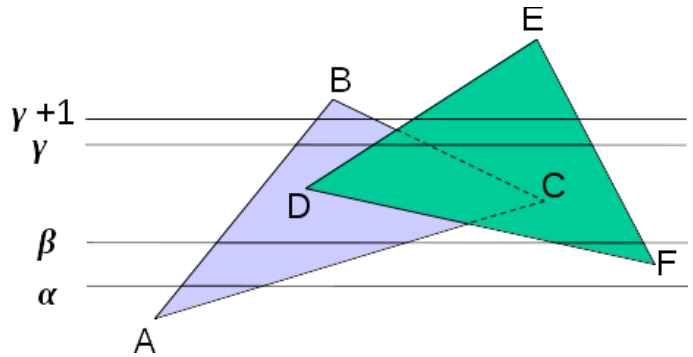
γ: $\underbrace{AB, AC}_{ABC}$ AB-nél `be-`, AC-nél `ki`kapcsolva

AB-től AC-ig ABC színe

$\gamma + 1$: $\underbrace{AB, AC}_{ABC}, \underbrace{FD, FE}_{DEF}$

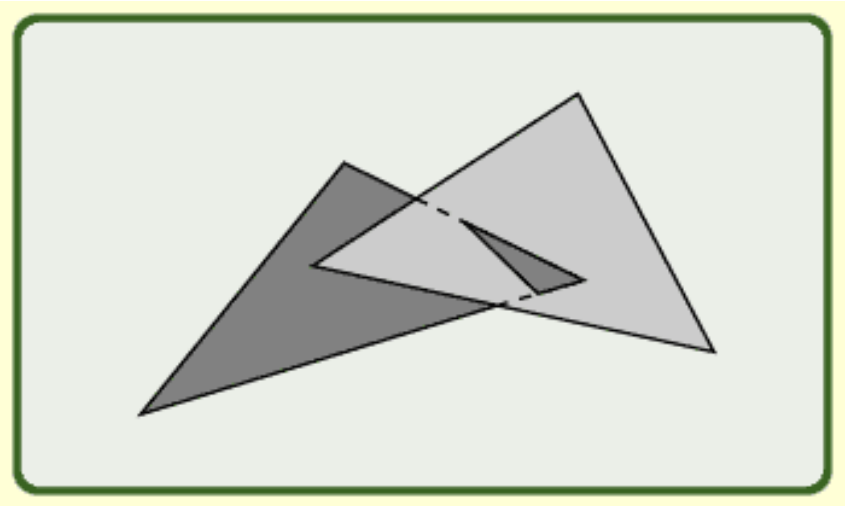
`be` `ki` `be` `ki`

Újab sorra térve **AÉT**-t rendezni kell!



A síkok egyenletéből dönthető el, hogy melyik van közelebb.

Átmetsző poligonok feldarabolása



Ha a poligonokat felvágjuk a metszeteik mentén, akkor nem kell minden pontban megvizsgálni a poligonok sorrendjét, elegendő csak akkor, ha egy „takaró” poligon véget ér.

Látható felszín beállítások OpenGL-ben

OpenGL-ben: a mélységi- vagy z-buffer algoritmus van implementálva.

A mélységbeli összehasonlítás engedélyezése és tiltása:

- `glEnable(GL_DEPTH_TEST)` // engedélyezés
- `glDisable(GL_DEPTH_TEST)` // tiltás

Sokszögek oldalai - OpenGL

Sokszögek (elülső és hátulsó) oldalai OpenGL-ben:
Elülső oldal az, amelyen a csúcspontok az óramutató járásával ellentétes irányban vannak megadva.

`void glPolygonMode(enum face, enum mode);` `face:`
`GL_FRONT_AND_BACK, GL_FRONT, GL_BACK`

Megmondja, hogy a poligon mindkét, vagy csak az elülső, vagy csak a hátulsó oldalát kell rajzolni.
`mode:` Megmondja a rajzolás módját

- `GL_POINT` csak a poligon csúcsait rajzolja
- `GL_LINE` határvonalát kell kirajzolni
- `GL_FILL` ki is kell tölteni (alapértelmezés)

Sokszögek oldalai - OpenGL

Elülső és hátulsó oldalak explicit megadása:

`glFrontFace(GLenum mode);`

`mode:`

- `GL_CW` az az oldal lesz elülső oldal, amelyen a csúcspontokat az óramutató járásával megegyező irányban adtuk meg,
- `GL_CCW` az ellenkezője

Sokszögek oldalai - OpenGL

A sokszög meghatározott oldalán letiltja a világítási, árnyalási és szín számítási műveleteket (láthatalan oldal)

`glCullFace(GLenum mode);`
`mode:  GL_FRONT, GL_BACK`
Ennek engedélyezése és tiltása:

- `glEnable(GL_CULL_FACE)` engedélyezés
- `glDisable(GL_CULL_FACE)` tiltás