1. **Fénysugár követő és festő algoritmus (3p)**

A **fénysugárkövető módszer** azt használja ki, hogy a kép pontokból (pixelekből) épül fel. Alapötlete az, hogy meghatározza az ábrázolandó objektumnak az a pontját, amely egy adott pixelen látszik.

Az algoritmus a következő lépésekből áll:

- a nézőpontot (a vetítés középpontját) összekötjük a kép pixeleivel, azaz egy sugarat bocsátunk ki a nézőpontból a pixelen-pontosabban, a képsíkon a pixeleknek megfelelő téglalap szimmetria- középpontján-át a modelltérbe.
- Ezekkel a félegyenesekkel elmetsszük az ábrázolandó objektumokat.
- Meghatározzuk a nézőponthoz legközelebbi metszéspontot, majd ennek a pontnak a színét, és a pixelhez ezt a színt rendeljük.

Ez az algoritmus, bár alapötletét tekintve egyszerű, sok számítást kíván, ezért időigényes. Az egyes pixelek színének meghatározása független egymástól, ezért az algoritmus alkalmas a párhuzamos feldolgozásra, amivel jelentősen felgyorsítható.

Festő algoritmus:

Ennek az algoritmusnak az alapgondolata, hogy a megjelenítendő objektumokat a nézőponttól való távolság függvényében sorba kell állítani. Az algoritmus a helyes takarási viszonyokat úgy alakítja ki, hogy a nézőhöz közelebb eső objektum képe felülírja a távolabbi objektum képét. Ennek legegyszerűbb változata az un. festő algoritmus, mely a képfestés technikájához hasonlóan a távolabbi objektumokra ráfesti a közelebbieket. Ezek az algoritmusok általában háromszögekkel való közelítését tételezik fel. Ha egy háromszög z irányban egyértelműen takarja a másikat, akkor a megjelenítésnél ennek képével felül kell írni a távolabbi háromszöget. A mélység rendező algoritmusokat általában valamilyen képpontosság eljárással együtt alkalmazzák. A pixeles megjelenítés pl. egy z-pufferrel kombinált sorkövető algoritmussal történhet.

Lépések:

- rendezzük a poligonokat a legkisebb (legtávolabbi pontjaik) z koordinátáik szerint.
- Hátulról előre haladva fessük ki a poligonokat.

Problémák: ciklikus átfedés, lapok kölcsönös átfedése

Gyorsítás: párhuzamosítás.

2. **Fények, fény, fény az openGL-ben, anyagtulajdonság (12pt)**

Az OpenGL csak a csúcspontokban számítja ki a szint. Egy csúcspont színe nem más, mint a pontból a nézőpontba (a szembe) eljutó fény színe, amit az objektum anyagának tulajdonságai, az uralkodó fényviszonyok, továbbá az ábrázolt alakzatok optikai kölcsönhatásai határoznak meg.

Fényösszetevők:

• **A környezeti fény** (ambient light) az ábrázolandó térrészben mindenütt jelen lévő, állandó intenzitású fény, melynek forrása, iránya nem ismert. A környezeti fény két részből tevődik össze:

- a térben a fényforrásoktól függetlenül jelen lévő környezeti fényből – globális környezeti fény.
- a fényforrásokból származó (pl. többszörös tükröződések) környezeti fényből – fényforrások környezeti fénykomponense

• **A szórt fénynek** (diffuse light) van iránya, mindig valamelyik fényforrásból jön. Fő jellemzője, hogy az objektumokkal ütközve minden irányba azonos módon és mértékben verődik vissza. Hatása nézőponttól független, csak a fényforrástól, az anyagtulajdonságoktól és a csúcspontbeli normálistól függ.

• **A tükrözött fénynek** (specular light) is van iránya és forrása, és hatása nemcsak az anyagtulajdonságoktól és a csúcspontbeli normálistól, hanem a nézőponttól is függ.

Fényforrás megadása:

- ***fényforrás helye*** Lehet végesben lévő, vagy végtelen távoli pont is.
- A végtelen távolban lévő fényforrásból kibocsátott sugarak párhuzamosak (napsugarak).
- A végesben lévő fényforrások (pontszerű fényforrások) azonos intenzitású fényt bocsátanak ki minden irányba. Megadható, hogy a fényforrástól távolodva milyen mértékben csökken a fény erőssége, azaz hogyan tompul a fény (attenuation). Reflektor (spot-light): Végesben lévő fényforrásból reflektort is létrehozhatunk. Reflektor létrehozásához meg kell adnunk a fénykúp tengelyének irányát és fél nyílásszögét, továbbá azt, hogy a fény intenzitása hogyan csökken a kúp tengelyétől a palást fele haladva.
- ***kibocsájtott fény*** A fényforrás által kibocsátott fény 3 összetevőből áll: környezeti fény (ambient light), szórt fény (diffuse light) és tükrözött fény (specular light). Mindhárom
színét az RGBA komponenseivel kell megadni.
- ***hozzájárulása a környezeti fényhez*** A környezeti fényösszetevő azt adja meg, hogy az adott fényforrás milyen mértékben járul hozzá a térrész környezeti fényéhez.

Anyagtulajdonságok:

Az objektumok anyagának optikai tulajdonságai:

• az objektum által kibocsátott fényt;
• az anyag milyen mértékben veri vissza a környezeti, a szórt es a tükrözött fényt;
• a ragyogást.

Fényvisszaverési tulajdonságok megadása: (tapasztalati mennyiségek)

• környezeti fény visszaverődési együttható (ambient reflection);
• szórt fény visszaverődési együttható (diffuse reflection);
• tükrözött fény visszaverődési együttható (specular reflection).

Ezek a konstansok tapasztalati mennyiségek, nem azonosíthatók a felület anyagának valamely fizikai jellemzőjével.

Egy felületi pontból a szembe jutó környezeti fény: $E_a = k_a * I_a$

Alakban írható fel, ahol I_a a megjelenített térrészben lévő összes környezeti fény, k_a pedig a felület környezeti fény visszaverődési együtthatója, komponensei a [0,1] intervallumbeli valós számok lehetnek.

3. Milyen színkeverési eljárásokat használnak a számítógépi grafikában? (4p)

A színeket legcélszerűbb egy kétdimenziós tér pontjaiként felfogni. A tér egy bázisának rögzítése után minden színt egy számhármas segítségével lehet azonosítani. A számítógépi grafikában két színkeverési mód terjedt el. Az egyik a színes grafikus megjelenítőknél használt additív (hozzáadó) színkeverés, a másik a színes nyomtatóknál használt szubtraktív (kivonó).

Additív színkeverés esetén azt írjuk elő, hogy mennyi vörös (red), zöld (green) és kék (blue) színkomponenst kell adni a fekete színhez, a fekete képernyőhöz. A színteret az egységkockával szemléltetjük, mivel a színkomponensek értéke általában a [0,1] intervallumon változtatható a különböző grafikus rendszerekben. Ezt a szín előállítási módot RGB színrendszernek szokás nevezni. A (0,0,0) számhármas a fekete, az (1,1,1) a fehér színnek felel meg. A színes raszteres grafikus megjelenítők működése ezzel összhangban van, hiszen a képernyő minden egyes pixele gyakorlatilag három

részből áll, melyek különböző intenzitású vörös, zöld és kék színt tudnak kibocsátani. Az RGB színmegadás az emberi szem működésével összhangban van.
Szubtraktív színkeverés esetén azt írjuk elő, hogy mennyi türkiz (cyan), bíbor (magenta) és sárga (yellow) színt kell kivonni a legösszetettebb szanból, a fehérből a kívánt szín előállítása érdekében. Ezt röviden CMY színrendszernek nevezzük.

A grafikai hardver által használt színkeverésen kívül más módok is lehetségesek, más bázis is felvehető a színtérben. Ilyen például a képzőművészek által használt színmeghatározás, mely a színárnyalat (Hue), fényesség (Lightness) és telítettség (Saturation) összetevőkön alapul. Ez a HLS színkeverési mód.

A számítógépek grafikus alaprendszereiben a színek megadása általában kétféleképpen történhet. Az egyik színmegadási mód esetén közvetlenül megadjuk a kívánt szín RGB komponenseit, a másik esetén csupán egy indexszel hivatkozunk egy színtáblázat (színpaletta) valamely elemére. Természetesen a paletta színei a felhasználó által megváltoztathatók, vagyis mi dönthetjük el, hogy az egyes színindexekhez milyen szín tartozzon és ezt bármikor megváltoztathatjuk.

Több rendszerben az RGB komponensek mellett egy negyedik, úgynevezett alfa (A) komponenst is megadhatunk, ezért ilyen esetben RGBA módról beszélünk. Az alfa komponens természetesen nem a szín megadásához kell, ahhoz pontosan három komponens szükséges, vele az átlátszóságot adhatjuk meg, vagyis azt kontrollálhatjuk, hogy az adott színnel kifestendő pixelen csak az új szín látsszon, vagy a pixel régi és új színéből együttesen álljon elő a pixel végső színe. Alfa 1 értéke esetén a szín nem átlátszó, 0 esetén teljesen átlátszó, a közbülső értékek esetén pedig a pixel régi és új színét keveri a rendszer az előírt arányban.

4. Milyen transzformáción mennek át a pontok, mire megjelennek (részletesen) (6p)
- **nézőpontba transzformálás**: A nézőpont és a nézési irány kiválasztásával azt határozzuk meg, hogy az alakzatot honnan és milyen irányból nézzük, vagyis azt, hogy az alakzat mely részét látjuk. Alapértelmezés szerint az origóból nézünk a negatív z tengely irányába. Ezzel un. nézőpontkoordináta-rendszert hozzuk létre.
- **modelltranszformáció:** Az ábrázolandó alakzatot elmozgathatjuk a koordináta-rendszerhez képest, vagy akár alakjukat és méretarányaikat is megváltoztathatjuk.
Modelltranszformációk: Egybevágósági transzformációk: bármely 2 pont és a neki megfelelő 2 pont távolsága egyenlő. eltolás, elforgatás, tükrözés, mozgatás (eltolás és forgatás)
Hasonlósági transzformációk: bármely 2 pont és a neki megfelelő 2 pont távolságának aránya állandó. kicsinyítés-nagyítás
Affin transzformáció: osztóviszony állandó. skálázás, nyírás
Projektív transzformáció: kettősviszony állandó.
- **vetítési transzformáció**: transzformáció, mely a teret síkra képezik le. A rendszer erre két lehetőséget kínál: a centrális és a merőleges vetítést. A vetítés módjának kiválasztásához a nézőpontkoordináta- rendszerben meg kell adnunk azt a térrészt, amit ábrázolni szeretnénk. Ez centrális vetítés esetén csonka gúla, merőleges vetítésnél téglatest. A vetítési transzformáció nem síkbeli képet eredményez, hanem a vetítendő térrészt (csonka gúlát, téglatestet) egy kockára képezi le. Centrális (perspektív) vetítést akkor alkalmazunk, ha valószerű képet szeretnénk létrehozni (az emberi látás is így működik), merőleges vetítést pedig méret- vagy méretarány-helyes képek létrehozásakor.
- **képmezőtranszformáció:** A képmező a képernyő ablakának az a téglalap alakú pixeltömbje, amire rajzolunk. A képmező-transzformáció a definiált kockát képezi le arra a téglatestre, melynek egyik lapja a képmező. Az (x, y) koordinátapárral a

képmező bal alsó sarkát, a width paraméterrel a szélességét, a height paraméterrel pedig a magasságát kell megadnunk pixelekben.

5. Z-puffer algoritmus:

A mélység-puffer (depth buffer, z-buffer) algoritmus segítségével poligonokkal határolt alakzatokat ábrázolhatunk színes raszteres megjelenítőkön. Az algoritmus azt vizsgálja, hogy egy-egy pixelen mi látszik, de nem kell rendezést végrehajtani pixelenként sem, csupán mindig két értéket kell összehasonlítani. Az algoritmushoz a pixelek színének tárolása (frame buffer) mellett meg kell oldani a mélységek (z értékek) tárolását is. Erre szolgál az úgynevezett mélység-puffer vagy z-puffer, ami egyben névadója is az algoritmusnak.

Az algoritmus főbb lépései a következők:

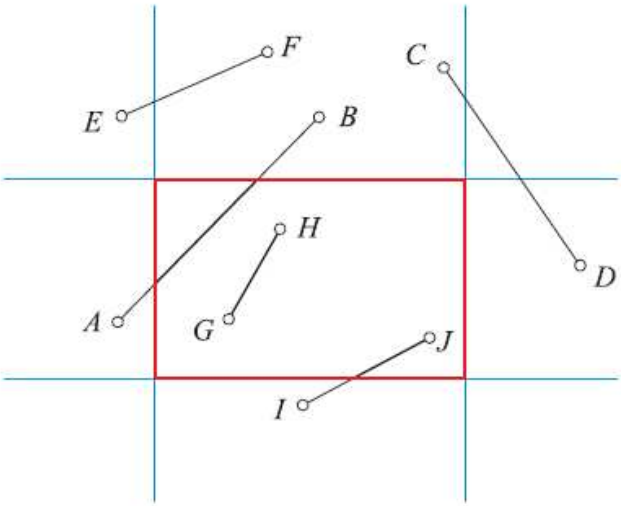
- a mélység-puffert a maximális mélységre állítjuk, vagyis arra a mélységre, amelytől távolabbi pontokat már nem veszünk figyelembe, a pixelekhez pedig a háttér színét rendeljük;
- az objektumok határoló felületén végighaladva, a határoló felületet pixelekké konvertálva, a felület képére eső minden pixelre:
  - meghatározzuk a pixelhez tartozó felületi pont mélységét;
  - ha ez az érték kisebb a pixelnek a mélység-pufferben tárolt értékénél, vagyis az új pont látszik, akkor:
    - a pixel mélységét a pont mélységére állítjuk be;
    - meghatározzuk a pont színét, és ezt rendeljük a pixelhez.

Bár ezt az algoritmust eredetileg poligonok, poligonhálók láthatóság szerinti ábrázolására fejlesztették ki - ugyanis ezeket lehet legegyszerűbben pixellé konvertálni -, minden olyan objektum láthatóság szerinti ábrázolására alkalmas, mely képének pontjaihoz (a képre eső pixelekhez) meghatározható a mélység és a szín. Poliédermodell esetén a felületi pontok meghatározása, a felület pixelekké konvertálása nem nehéz feladat.

Az algoritmus egyszerű, létezik hardver implementációja is, amit napjainkban egyre több grafikus kártya tartalmaz. Az algoritmus hátránya a mélység-pufferhez szükséges viszonylag nagy memóriaterület.

6. Chonen-Sutherland algoritmus:

A Cohen-Sutherland algoritmus a képsíkot a képmező oldalegyeneseivel 9 tartományra osztja. Az egyes tartományokhoz kódot rendel, aminek tárolásához tehát 4 bit elegendő. A képmező kódja 0000, a többi tartományhoz úgy rendel kódot, hogy minden oldalegyenest hozzárendel egy bithez, és azt a bitet 0-ra állítja, ha a tartomány az oldalegyenesnek a képmező felé eső oldalán van, egyébként 1-re állítja (lásd a 3.2. ábrát). Az algoritmus lépései a következők:



3.3. ábra. Különböző helyzetű szakaszok

1. A vizsgált szakasz kezdő- és végpontjának meghatározzuk a kódját, és ezeket az a és b változókba tesszük.

2. Ha $a \mid b = 0$ ($\mid$ a bitenkénti "vagy" műveletet jelöli), akkor a szakasz vágás nélkül megrajzolható.

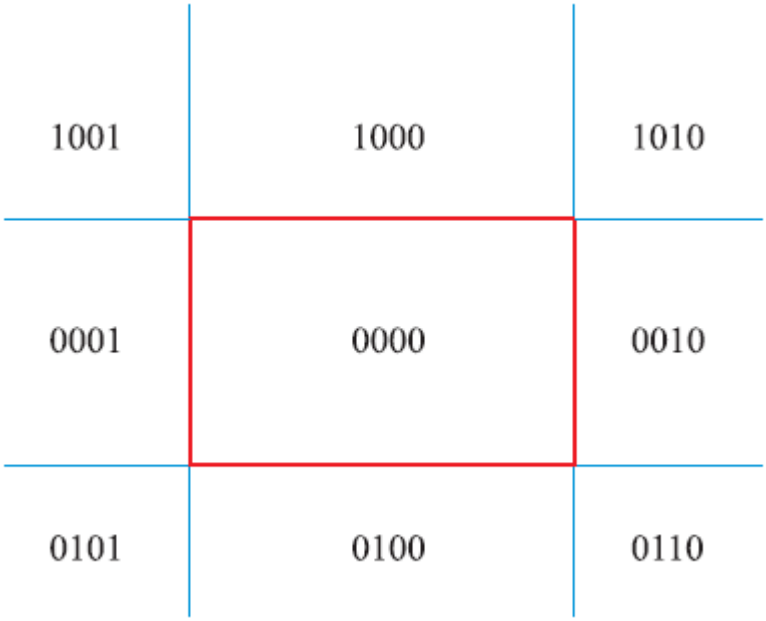
3. Ha $a \,\&\, b \,!= 0$ ($\&$ a bitenkénti "és" műveletet jelöli), akkor a szakasz elhagyható, mert egyetlen pontja sem látszik, mivel a kezdő- és végpontja is valamelyik oldalegyenesnek a képmezővel ellentétes oldalán van.

4. Megkeressük az első olyan oldalegyenest, amelyik metszi a szakaszt, a metszésponttal helyettesítjük a megfelelő végpontot, az új végpontnak meghatározzuk a kódját és az eljárást a 2. lépéstől újrakezdjük.

A fenti algoritmussal legfeljebb két vágás után a 2. vagy 3. lépésbeli feltétel teljesül. Két vágás a 3.3. ábra szerinti AB és CD szakaszok esetén szükséges. Előfordulhat, hogy csak a második vágás után derül ki a szakaszról (az ábrán a CD szakasz), hogy egyetlen pontja sem látszik. A GH szakaszról azonnal kiderül, hogy nem kell vágni, az EF-ről pedig az , hogy elhagyható. Az IJ szakaszt egyszer kell vágni.

A poligonok vágását nem lehet leegyszerűsíteni a határoló szakaszok vágására. A poligon ugyanis zárt síkrészt képvisel, belseje is van, tehát a vágás után is zárt síkidomot kell kapnunk. (Egyébként nem tudnánk a vágás után a belsejét kifesteni, szétfolyna a festék.)

A Sutherland-Hodgman poligonvágó algoritmus végigmegy a képmező oldalain, és az oldalegyenesekkel elvágja a az ábrázolandó poligont úgy, hogy minden egyes vágás után lezárja a csonkolt poligont az oldalegyenes megfelelő szakaszával (szakaszaival). Ezt illusztrálja a 3.4. ábra.



7. Árnyalási módok

Konstans árnyalás (flat shading) esetén minden lapnak csak egyetlen pontjában számítjuk ki a színt, és az egész lapra ezt alkalmazzuk. Csak akkor megoldás, ha a

fénysugarak párhuzamosak; a nézőpont végtelen távoli pont és az ábrázolt objektum poliéder.
Folytonos árnyalás (smooth shading) esetén a csúcspontok színét –RGBA módban a színt komponensenként, színindex módban az indexeket – lineárisan interpolálja a rendszer a szakaszok belső pontjai színének kiszámításához. Folytonos árnyalás alkalmazásával el tudjuk tüntetni a képről a görbült felületeket közelítő poligonháló éleit.
Gouroud-féle árnyalás: a csúcspontokban a csúcspontbeli normálisok segítségével kiszámítjuk a színt. A lap élei mentén a csúcspontok színének lineáris interpolációjával határozzuk meg a színt, a lap belső pontjaiban pedig kettős lineáris interpolációval. Hátránya, hogy a test körvonala nem lesz sima; Mach-sáv hatás (megoldás: modell finomítása).
Phong-féle árnyalás: nem a fény intenzitását interpolálja,
hanem a normálisokat. Hátránya a képhatár szögletessége.

8. Konvex ponthalmaz, konkáv ponthalmaz, konvex halmaz burka, egyszerű sokszög

Egy ponthalmazt konvexnek nevezünk, ha bármely két pontját összekötő szakasz minden pontja a ponthalmazhoz tartozik. A nemkonvex ponthalmazokat konkávnak nevezzük. A síkbeli konvex poligonok belső szögei 180 foknál nagyobbak, azaz a poligon bármely pontjából az össze többi pontot láthatjuk.

Akárhány konvex halmaz metszete konvex, így a vektorterek konvex részhalmazai hálót alkotnak. Ez azt is jelenti, hogy a vektorterek minden részhalmazának van **konvex burka**, azaz van egy őt tartalmazó legszűkebb konvex halmaz, ami az összes tartalmazó konvex halmaz metszete.

A geometriában **sokszögnek** (idegen szóval: **poligon**nak) nevezzük azokat a síkidomokat, melyeket véges sok egyenes szakasz alkotta zárt görbe (azaz zárt törött vonal) határol.

Ezeket a szakaszokat éleknek vagy oldalaknak nevezzük, és azokat a pontokat, ahol az élek találkoznak, csúcsoknak. A sokszögek belsejét *test*-nek nevezzük.

Konvex ponthalmaz: Ha az alakzat bármely két pontjával együtt a két pontot összekötő szakasz valamennyi pontját is tartalmazza.

Konkáv ponthalmaz: Ha van olyan az alakzat két pontjait összekötő szakasz, amelyik nem tartozik teljes egészében az alakzathoz.

Konvex halmaz burka: A legszűkebb konvex polinom, mely vagy belső pontként, vagy csúcspontként tartalmazza a halmaz összes pontját.

Egyszerű sokszög: Nem keresztezheti önmagát. (a benne szereplő szakaszoknak az előírt csatlakozási pontokon kívül nincs más közös pontjuk).

9. Terület felosztásos algoritmus

A terület felosztó algoritmusok a bonyolultabb takarási vizsgálatokat a kép területének kisebb részekre való rekurzív felosztásával vezetik vissza az egyszerűen kezelhető esetekre. A képfelosztás történhet a raszteres képernyő koordináta rendszerben, illetve

a vektoros látótér koordináta rendszerben. A legismertebb algoritmus ebben a csoportban Warnock algoritmusa.

A rekurzió után maradó lehetséges helyzetek:

a) minden poligon a vizsgált területen kívül esik.

b) Csak egy metsző, vagy tartalmazott poligon van a vizsgált területben. Először a területet kijelöljük háttérszínnel, majd a poligon területen belüli részét scan-konvertáljuk.

c) csak egy, a vizsgált területet teljesen tartalmazó poligon van. Befestjük a területet a poligonnak megfelelő színnel.

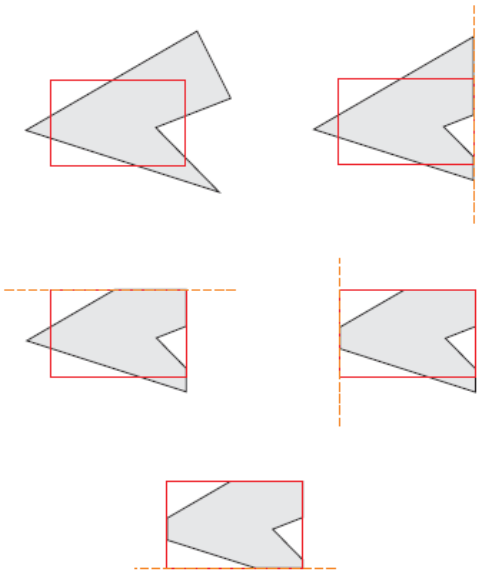
d) Több, mint egy poligonnak van közös része a tartománnyal, de van olyan, tartományt teljesen tartalmazó poligon, amely a többi előtt van. A rekurzív felosztás egybevágó téglalapokra vagy egy alkalmas belső pont választásával négy nem egybevágó téglalapra történhet. Ez utóbbi esetben a pont alkalmas választásával az algoritmus hatékonysága jelentősen növelhető.

10. Scan line algoritmus (pixelsor algoritmus)

A scan line algoritmusok pixelsoronként készítik a képet. Az algoritmus a jelenetet felépítő objektumokról gyűjtött információkat táblázatokba tárolja: a képsíkra vetített, nem vízszintes éleket az éltáblázat, a poligonok fontosabb paramétereit a poligon táblázat, az éppen vizsgált pixelsorhoz tartozó éleket az aktív éltáblázat tartalmazza. Egy-egy pixelsor vizsgálatakor az él táblázat azok az elemei, melyek metszik a sort, áttöltődnek az aktív éltáblázatba. A pixelsor és a háromszögek közös részét szegmenseknek nevezzük. Ha a pixelsor egy szegmense csak egy háromszöghöz tartozik, akkor ezt egyszerűen csak meg kell jeleníteni. Ha egy pixel több szegmenshez tartozik, akkor a megfelelő háromszögek mélységi vizsgálatával kell eldönteni, hogy melyik háromszög felületi pontját kell kirajzolni. Előnye: felesleges lépések kiszűrhetőek, gyorsítás: párhuzamosítással.

11. Sutherland – Hodgman poligonvágó algoritmus

Végigmegy a képmező oldalain, és az oldal egyenesekkel elvágja a poligont úgy, hogy minden egyes vágás után lezárja a csonkolt poligont az oldal egyenes megfelelő szakaszával.



Hátsó lapok eltávolítása (back-face culling) Poliéderek ábrázolásánál célszerű használni.

- ha a normálissal bezárt szög 90° élben látjuk

- ha 90°-nál nagyobb: biztosan nem látható

- ha 90°-nál kisebb: van esély hogy látható, de takarásban is lehet. Konvex poliéder esetén biztosan látható. Konkáv poliéder esetén lehet nem. Zárt felület esetén is csak akkor eredményez korrekt, láthatóság szerinti képet, ha egyetlen konvex poliédert ábrázolunk. Konkáv vagy több egymást részben átfedő poliéder esetén további vizsgálat szükséges. Ne használjuk nyílt poligonháló esetén.