RIEŠENIE VIDITEĽNOSTI V POČÍTAČOVEJ GRAFIKE

doc. Ing. Branislav Sobota, PhD.
Katedra počítačov a informatiky, FEI TU v Košiciach

VRSTVY VIZUALIZAČNÉHO PROCESU

1. Definovanie/spracovanie modelu (reprezentácia, súradnicové systémy)
2. Transformácie nad objektami
3. Riešenie viditeľnosti
4. Tieňovanie
5. Osvetľovanie
6. Realistické zobrazovanie
7. Kompozícia a Vykresľovanie

RIEŠENIE VIDITEĽNOSTI (VISIBILITY)

Spočíva v odstránení (resp. odlíšení) tých častí trojrozmerných objektov, ktoré pri danom premietaní do 2D nie sú z miesta pozorovateľa viditeľné. Tým sú tieto časti akože zakryté a dostávame jednoznačné priemety želaných telies.

RIEŠENIE VIDITEĽNOSTI - KATEGORIZÁCIA

- podľa priestoru, kde je viditeľnosť riešená:
  - riešenie v *3D*
  - riešenie v *2D* priemetne
- podľa reprezentácie objektov, ktorých viditeľnosť riešime:
  - *objektovo orientované algoritmy* (kde sa rieši, ktorá časť príslušného objektu je viditeľná)
  - *obrazovo orientované algoritmy* (kde sa rieši spätne pre každý obrazový bod, ktorý objekt je v ňom vidieť)

RIEŠENIE VIDITEĽNOSTI - KATEGORIZÁCIA

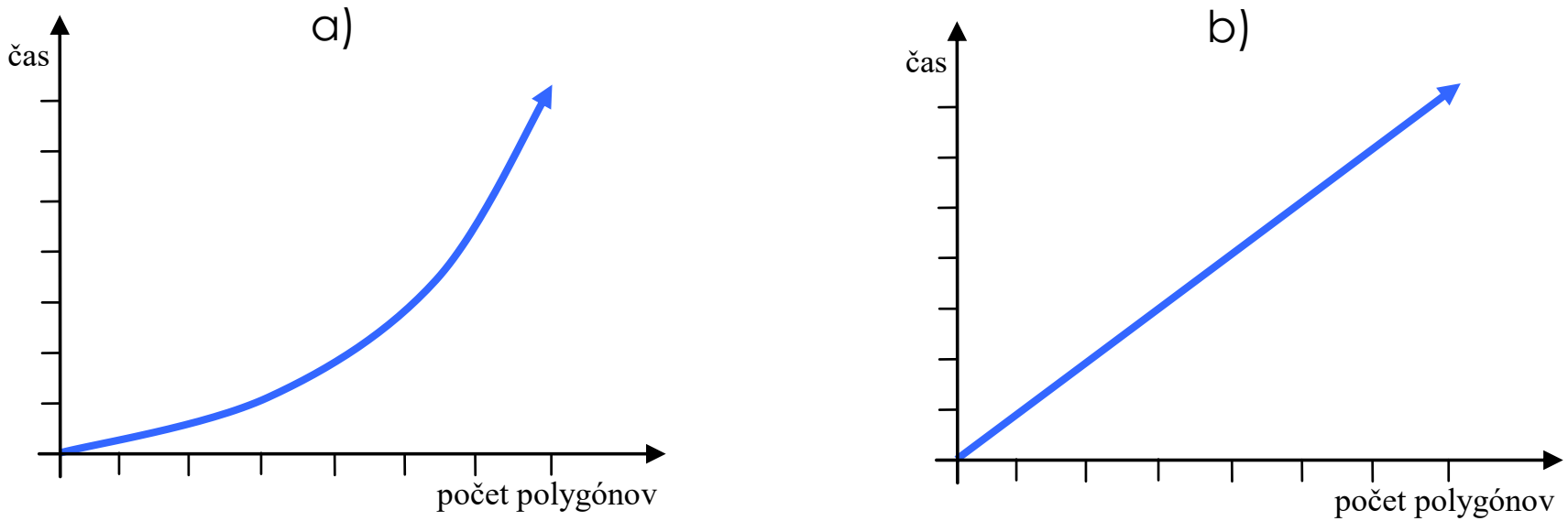
- podľa toho či sa uvažuje aj osvetlenie telesa:
  - *riešenie bez osvetlenia* (vyhodnotenie farieb je aplikované lokálne na každý objekt)
  - *riešenie s osvetlením* (medzi tieto radíme v podstate aj metódu sledovania lúča (raytracing) alebo vyžarovaciu metódu (radiosity), ktoré sa častokrát označujú aj ako globálne metódy riešenia viditeľnosti s globálnou aplikáciou farieb v rámci scény t.j. napr. aj odrazy)

RIEŠENIE VIDITEĽNOSTI - KATEGORIZÁCIA

- podľa vplyvu možnej chyby pri vykonávaní (napr. použitím celočíselnej aritmetiky):
  - *s lokálnym vplyvom chyby* na výsledok
  - *s globálnym vplyvom chyby* na výsledok
- podľa času potrebného na riešenie viditeľnosti:
  - riešenie *mimo reálneho času*
  - riešenie *v reálnom čase*

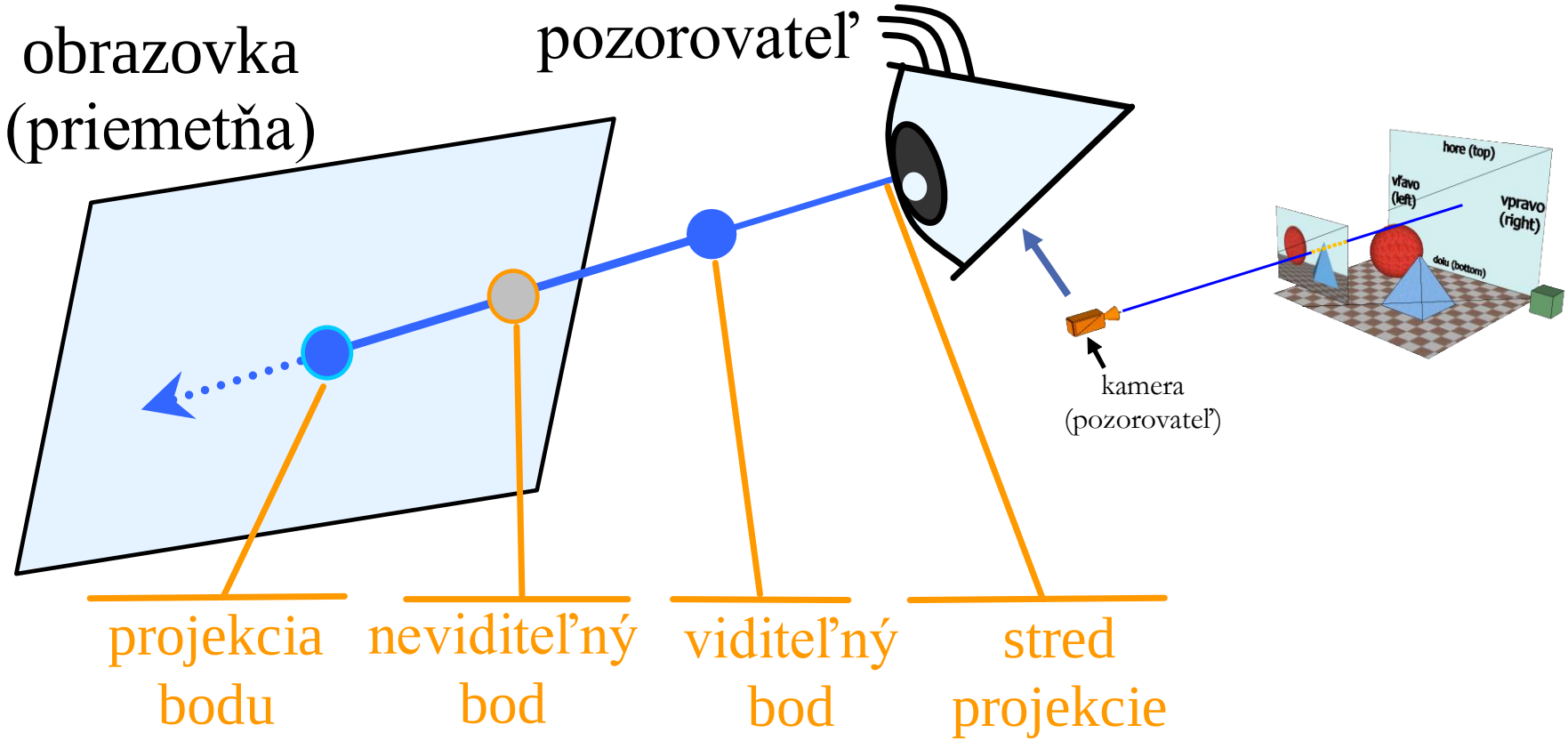
ČASOVÁ ZÁVISLOSŤ RIEŠENIA VIDITEĽNOSTI

$$T = T_{V_{ýber}} + T_{K_{reslenie}}$$



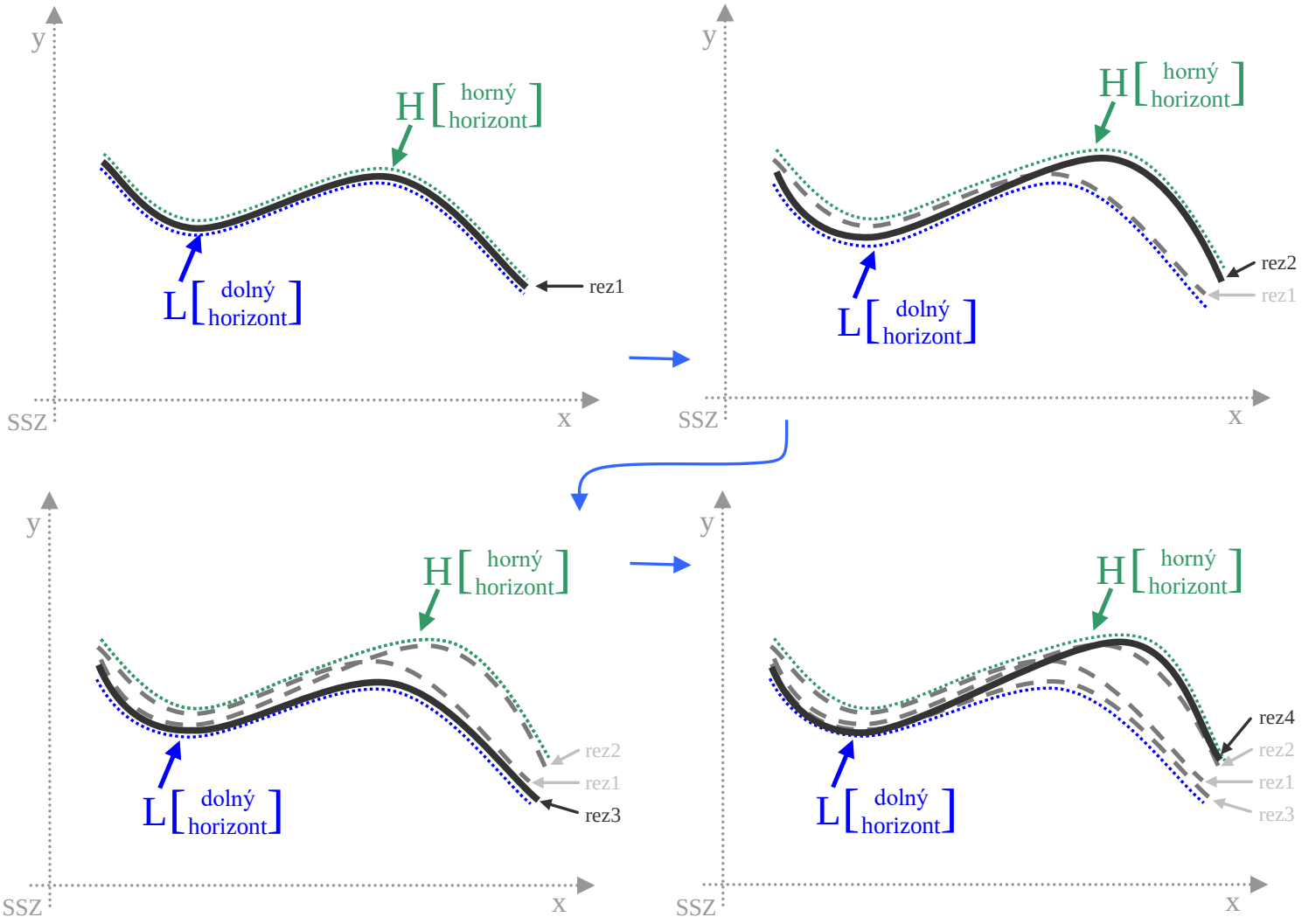
T_v pre a) objektové a b) rastrové algoritmy

VIDITEĽNOSŤ BODU



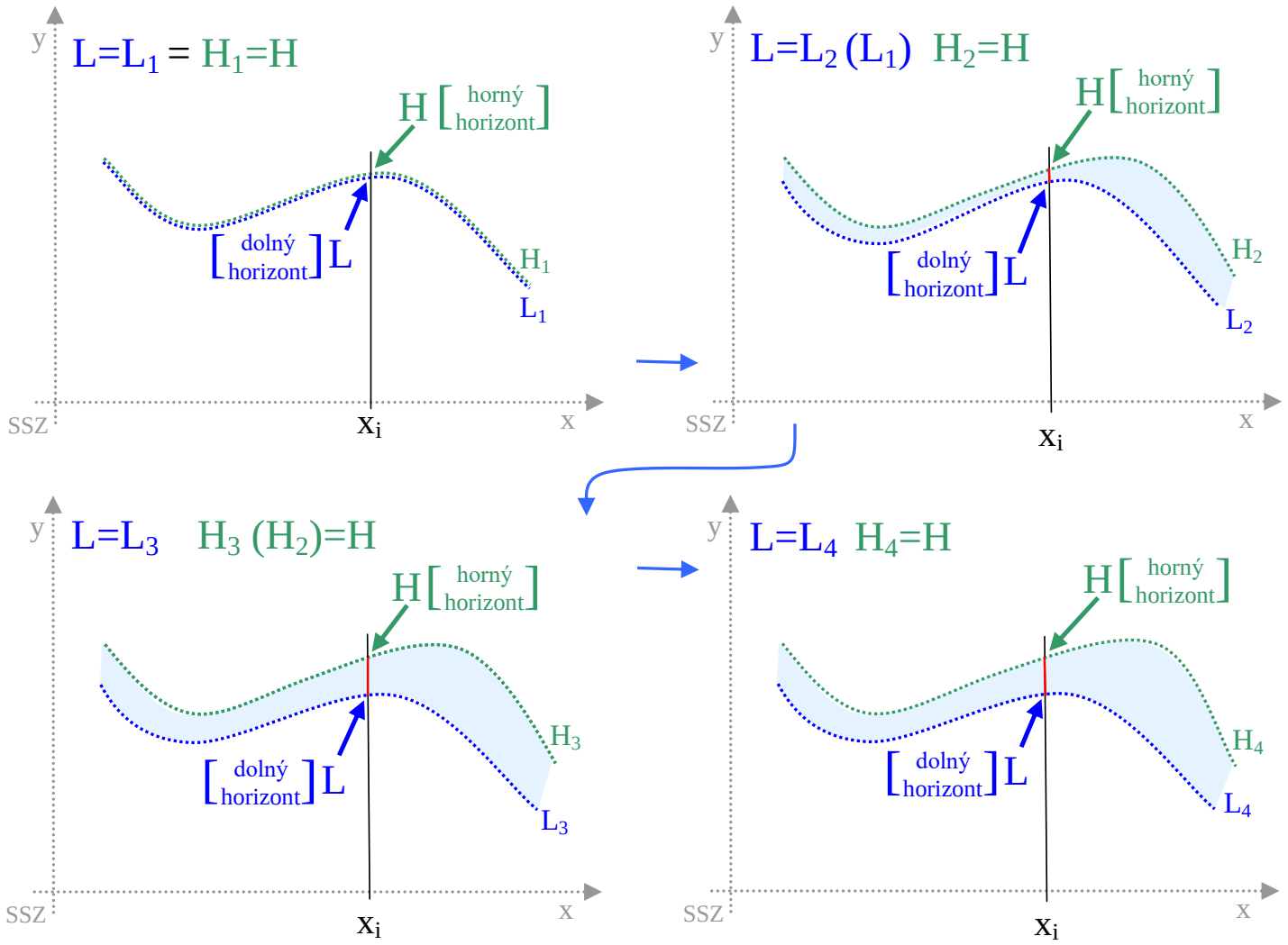
ALGORITMUS PLÁVAJÚCEHO HORIZONTU

NA ÚROVNI REZOV/KRIVIEK

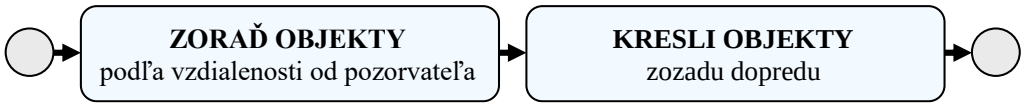


ALGORITMUS PLÁVAJÚCEHO HORIZONTU

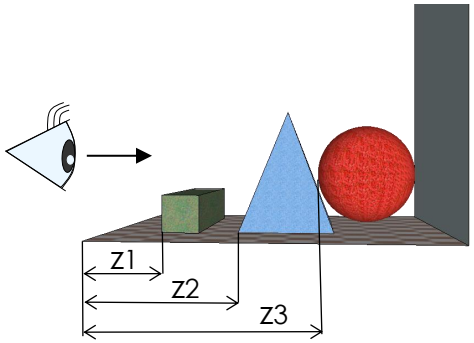
NA ÚROVNI INTERVALU HODNÔT V X_i



MALIAROV ALGORITMUS
PAINTER'S ALGORITHM



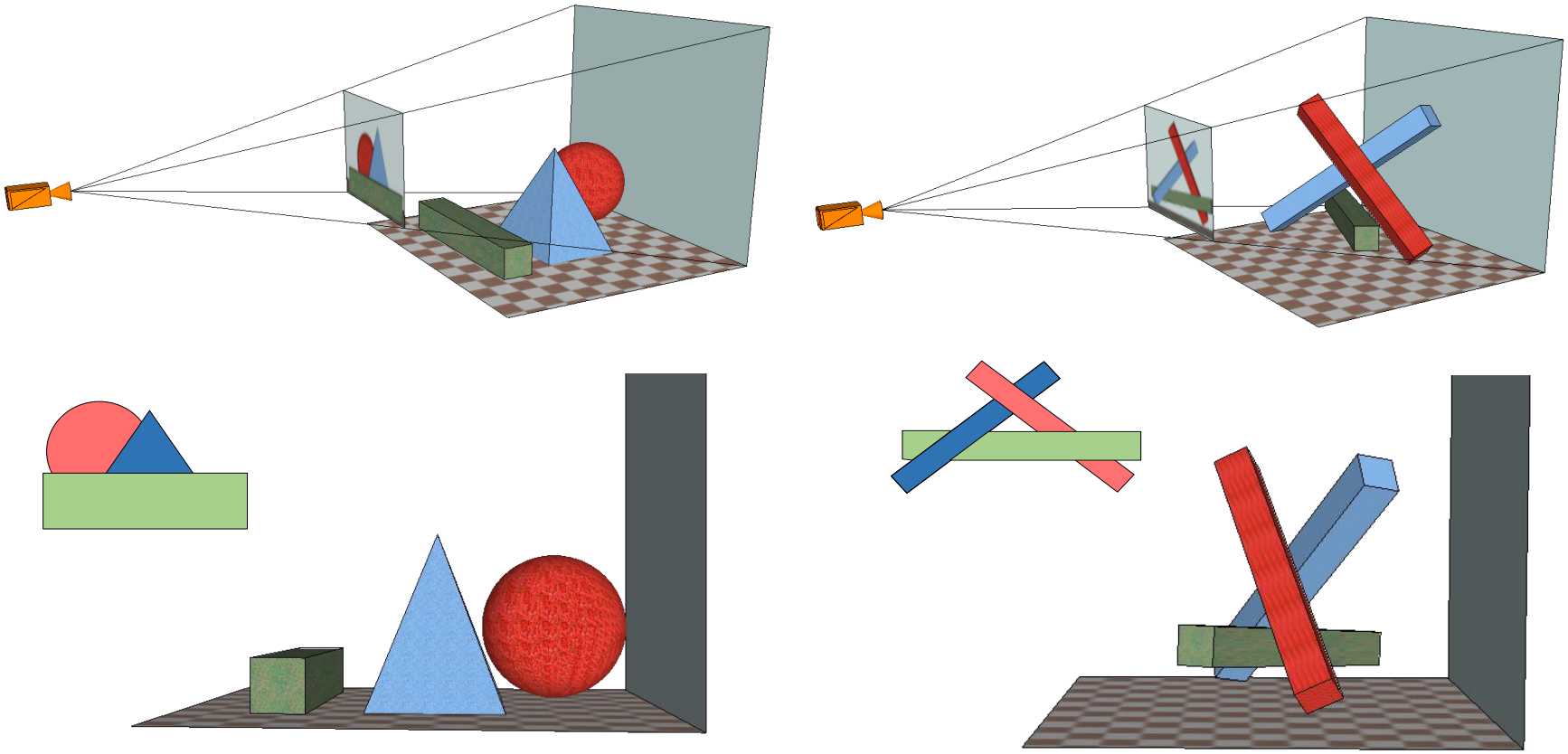
(Z-Sort, O(n.log2 n) v najlepšom prípade)



Podmienky:

- Ak $z1_{max} < z2_{min}$, potom je polygón *P1* úplne za polygónom *P2*.
- Ak sa neprekrývajú priemety polygónov v rovine priemetne (t.j. v *SSZ*), je poradie ich vykresľovania nezávislé a polygón P1 testu vyhovuje.
- Ak sú všetky vrcholy polygónu *P1* pod rovinou určenou polygónom *P2*, potom *P1* leží za polygónom *P2*.
- Ak sú všetky vrcholy polygónu *P2* nad rovinou určenou polygónom *P1*, potom *P1* leží za polygónom *P2*.

MALIAROV ALGORITMUS



Modelové scény pre maliarov algoritmus bez a s cyklickým prekryvom

WARNOCKOV ALGORITMUS

Warnockov algoritmus umožňuje zobraziť viditeľné časti telesa dvoma spôsobmi:

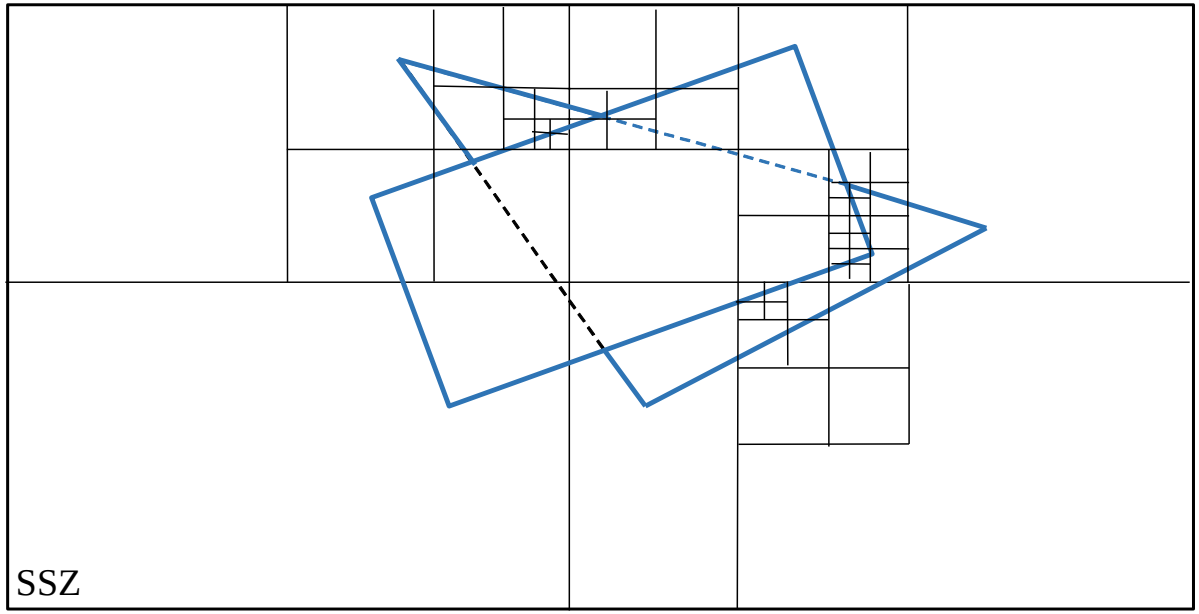
- hrany (drôtové zobrazenie)
- steny (plošné zobrazenie)

WARNOCKOV ALGORITMUS

1. Uloženie základného zobrazovacieho priestoru, teda jednej z tých 4.častí do zásobníka.
2. Výber zo zásobníka a sprístupnenie informácií, ktoré definujú veľkosť okna.
3. Ohodnotenie a rozhodnutie, vzhľadom na jednotlivé možnosti, ktoré môžu nastať pri kreslení telies po delení v príslušnom obdĺžniku (okne). Možnosti:
   a) V okne sa nachádza mnohouholník prekrývajúci daný obdĺžnik.
   b) Mnohouholník je mimo obdĺžnika.
   c) Ostatné prípady.
4. Keď sa jedná o možnosti a) alebo b), potom vieme jednoznačne rozhodnúť o viditeľnosti (výpočet vzdialenosti a rozhodnutie je v 3D). Nastáva vykreslenie.
5. Keď nevieme rozhodnúť o viditeľnosti, teda jedná sa o možnosť c), potom sa robí ďalšie (vlastne rekurzívne) delenie príslušného okna na 4 rovnaké časti. Vypočítajú sa súradnice jednotlivých okien a každé okno sa uloží do zásobníka.

WARNOCKOV ALGORITMUS

6. Keď rozmery okna sú rovné jednej, tzn. že sa jedná o bod (pixel), dá sa jednoznačne rozhodnúť, ktorý bod je viditeľný a ten sa vykreslí.
7. Keď sa zásobník vyprázdni, ukončí sa celý algoritmus.
8. V opačnom prípade sa pokračuje v kroku 2.



Postup delenia pri Warnockovom algoritme

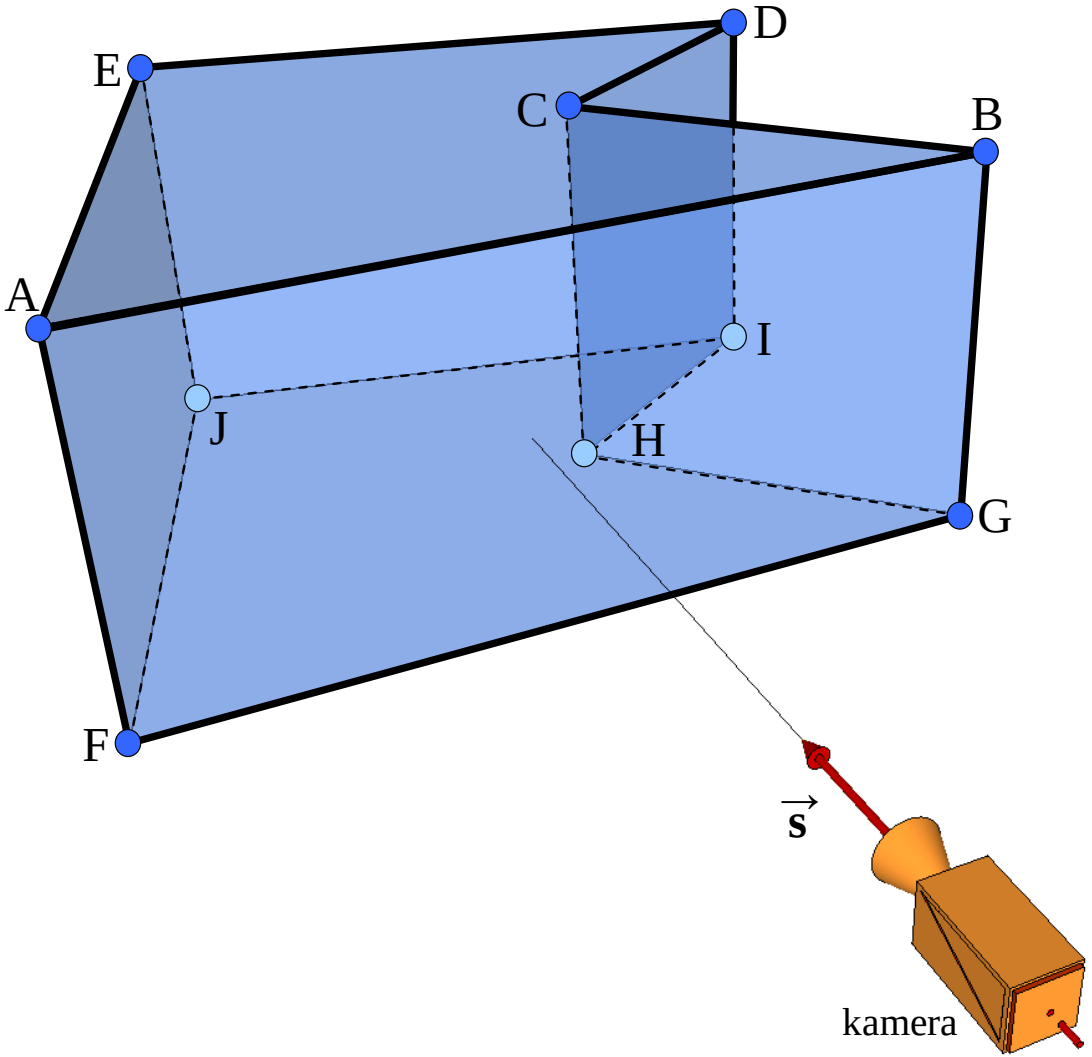
FREEMAN-LOUTRELOV ALGORITMUS

Algoritmus pracuje v 3D a je založený na rozdelení stien na neviditeľné a potencionálne viditeľné na základe uhla φ medzi vektorom pohľadu kamery $\vec{k}$ a normálou steny $\vec{n}$.

$$\varphi = \arccos\left(\frac{k_x \cdot n_x + k_y \cdot n_y + k_z \cdot n_z}{\sqrt{k_x^2 + k_y^2 + k_z^2} \cdot \sqrt{n_x^2 + n_y^2 + n_z^2}} \right)$$

Je potrebné zabezpečiť rovnakú orientáciu všetkých stien v scéne (v smere alebo proti smeru hodinových ručičiek)

FREEMAN-LOUTRELOV ALGORITMUS



Príklad typu stien:

orientácia

$\vec{n}(U,V,X,Z)$

normála príslušnej steny

$\vec{n}(B,C,H,G)\times\vec{s}>0$

zadná (neviditeľná)

$\vec{n}(C,D,I,H)\times\vec{s}<0$

predná
(potencionálne viditeľná)

Podmienka:

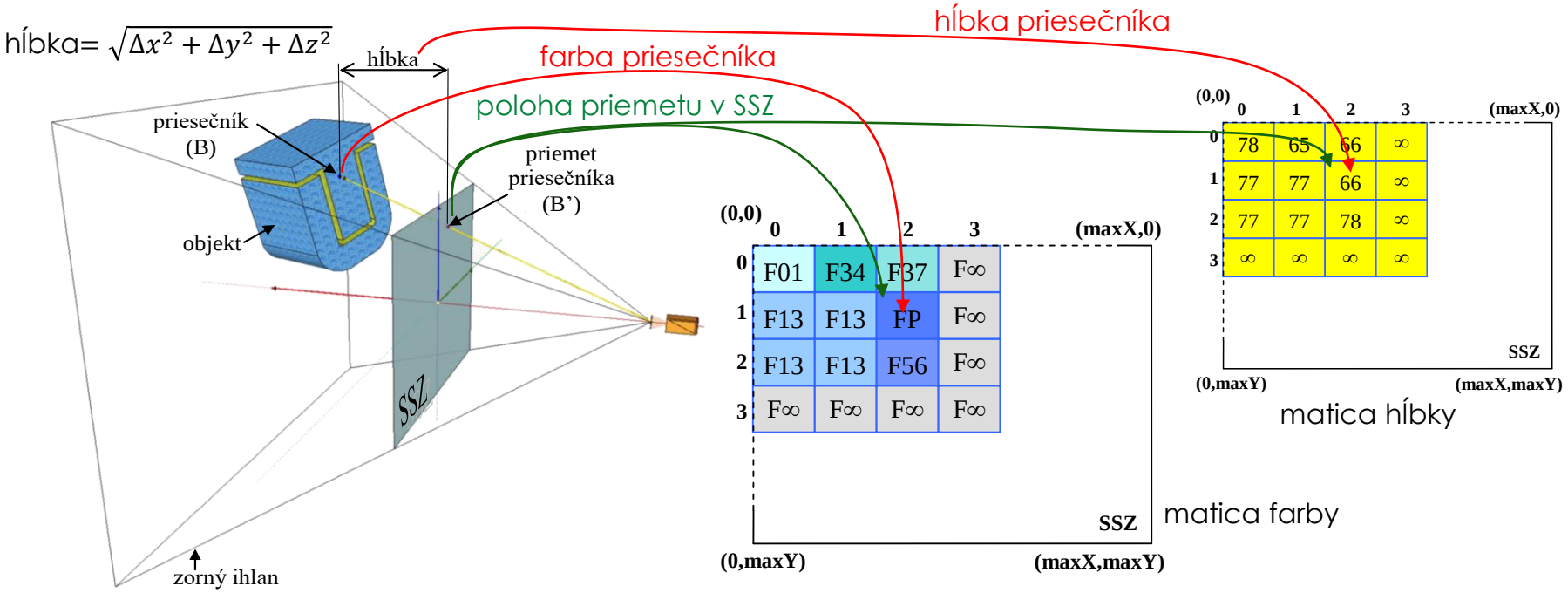
orientácia v celej scéne rovnaká

ALGORITMUS PAMÄTE HĹBKY (Z-BUFFER)

všeobecná rovnica roviny mnohouholníka je:

$$a.x + b.y + c.z + d = 0.$$

$$z_i = \frac{-a}{c} \cdot (x_i - x_0) + \frac{-b}{c} \cdot (y_i - y_0) + z_0$$



ALGORITMUS PAMÄTE HĹBKY (Z-BUFFER)

výhody:

- korektne rieši viditeľnosť
- polygóny (mnohouholníky) sa môžu navzájom pretínať
- polygóny je možné kresliť v ľubovoľnom poradí (nevyžaduje žiadne predspracovanie ani triedenie polygónov)
- vhodný pre objekty s množstvom malých polygónov

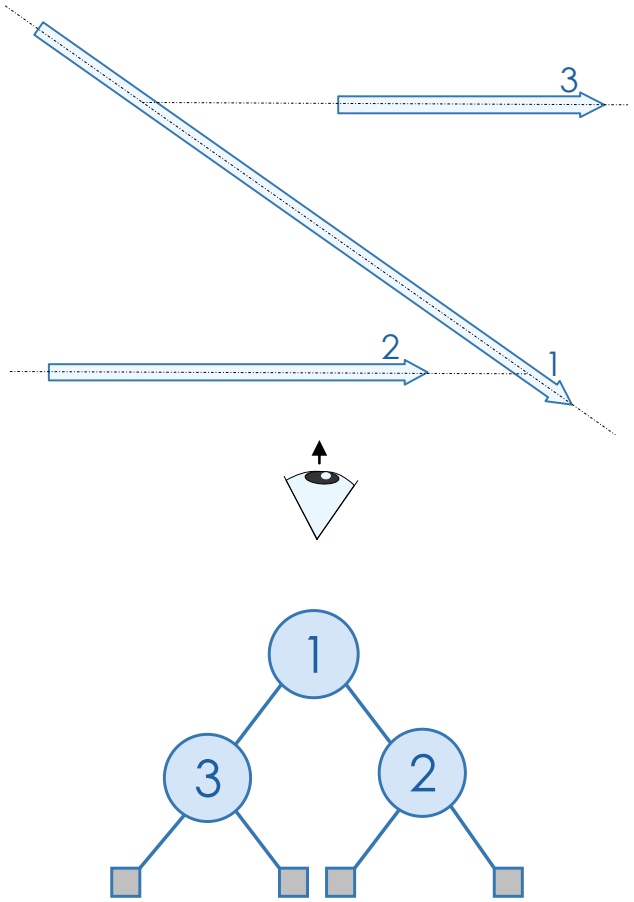
nevýhody:

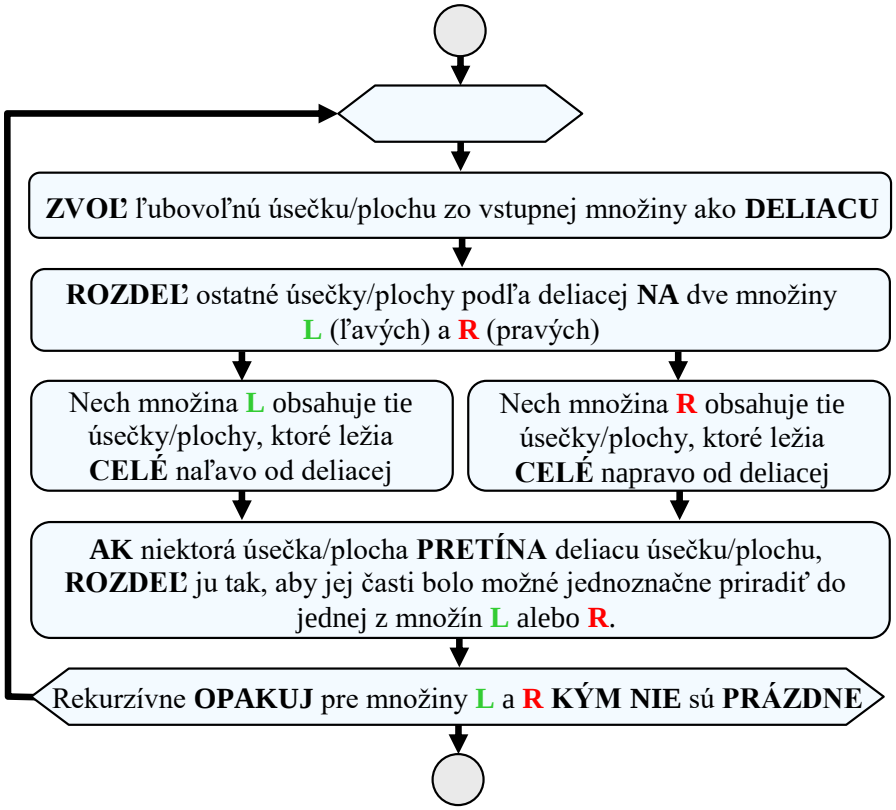
- vysoké nároky na pamäť, minimálne (2.maxX .maxY) bajtov
- prekresľovanie
- je pomalý (málo efektívny) pri veľkých scénach (veľa veľkých polygónov), je ho možné však urýchliť použitím orezania na zorný ihlan.

BSP stromy (Binary Space Partitioning)

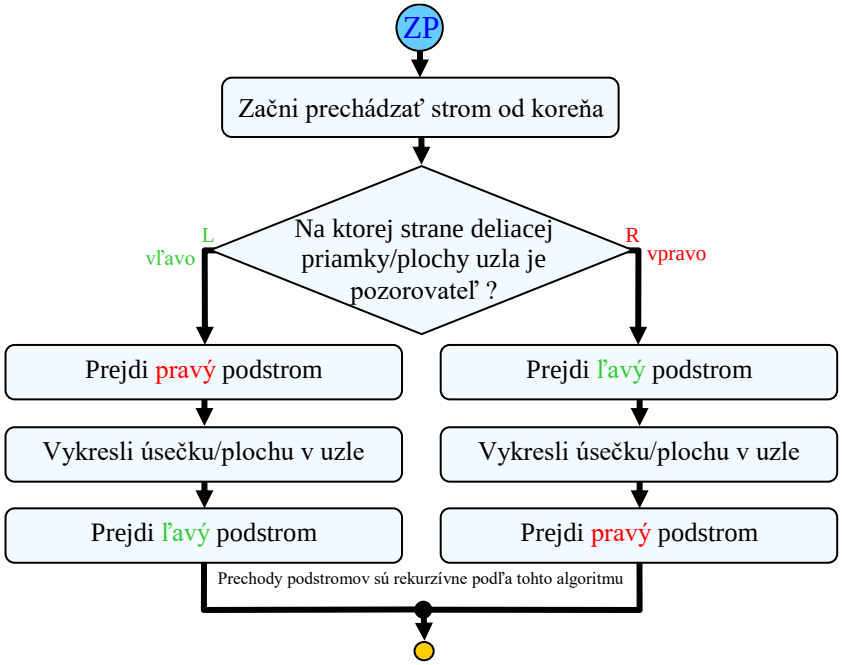
- Podľa rozloženia objektov:
  - Vyvážené
  - Nevyvážené
- Podľa rozloženia deliacich rovín:
  - Statické
  - Dynamické
  - Adaptívne
  - Stratové

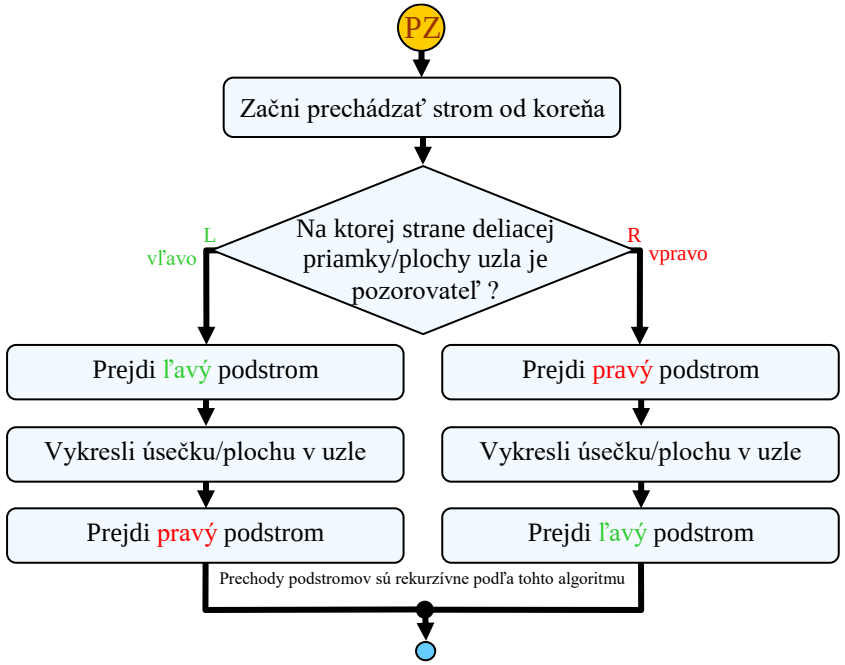
BSP STROMY


3
2
1
1
3
2

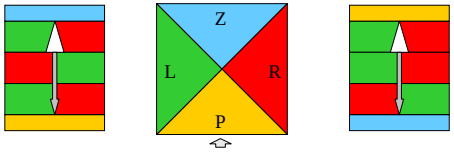

ZVOĽ ľubovoľnú úsečku/plochu zo vstupnej množiny ako DELIACU
ROZDEĽ ostatné úsečky/plochy podľa deliacej NA dve množiny L (ľavých) a R (pravých)
Nech množina L obsahuje tie úsečky/plochy, ktoré ležia CELÉ naľavo od deliacej
Nech množina R obsahuje tie úsečky/plochy, ktoré ležia CELÉ napravo od deliacej
AK niektorá úsečka/plocha PRETÍNA deliacu úsečku/plochu, ROZDEĽ ju tak, aby jej časti bolo možné jednoznačne priradiť do jednej z množín L alebo R.
Rekurzívne OPAKUJ pre množiny L a R KÝM NIE sú PRÁZDNE

BSP STROMY - PRECHOD STROMAMI



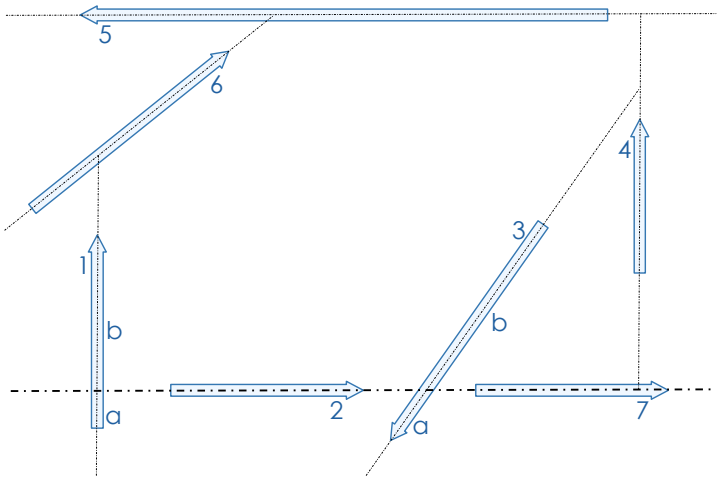


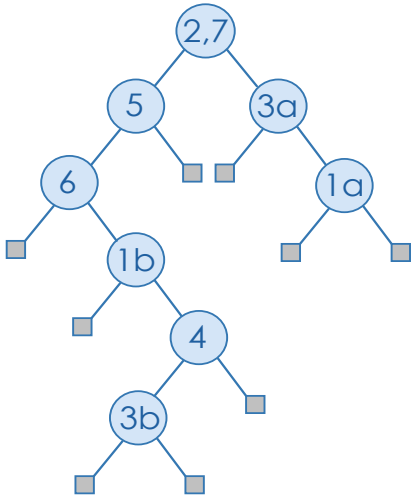
prechod zozadu dopredu

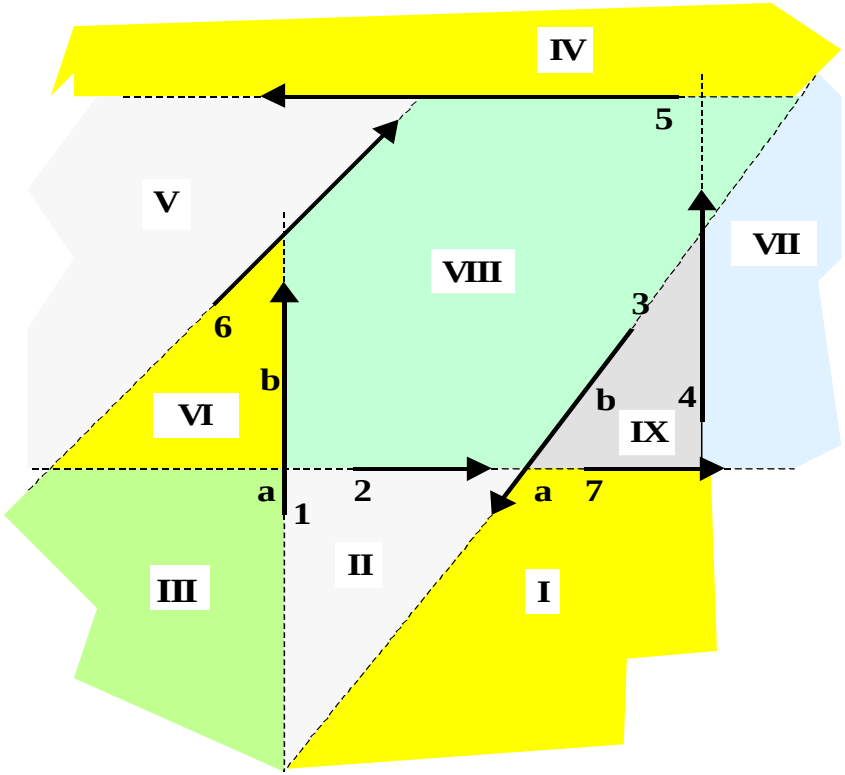


prechod spredu dozadu

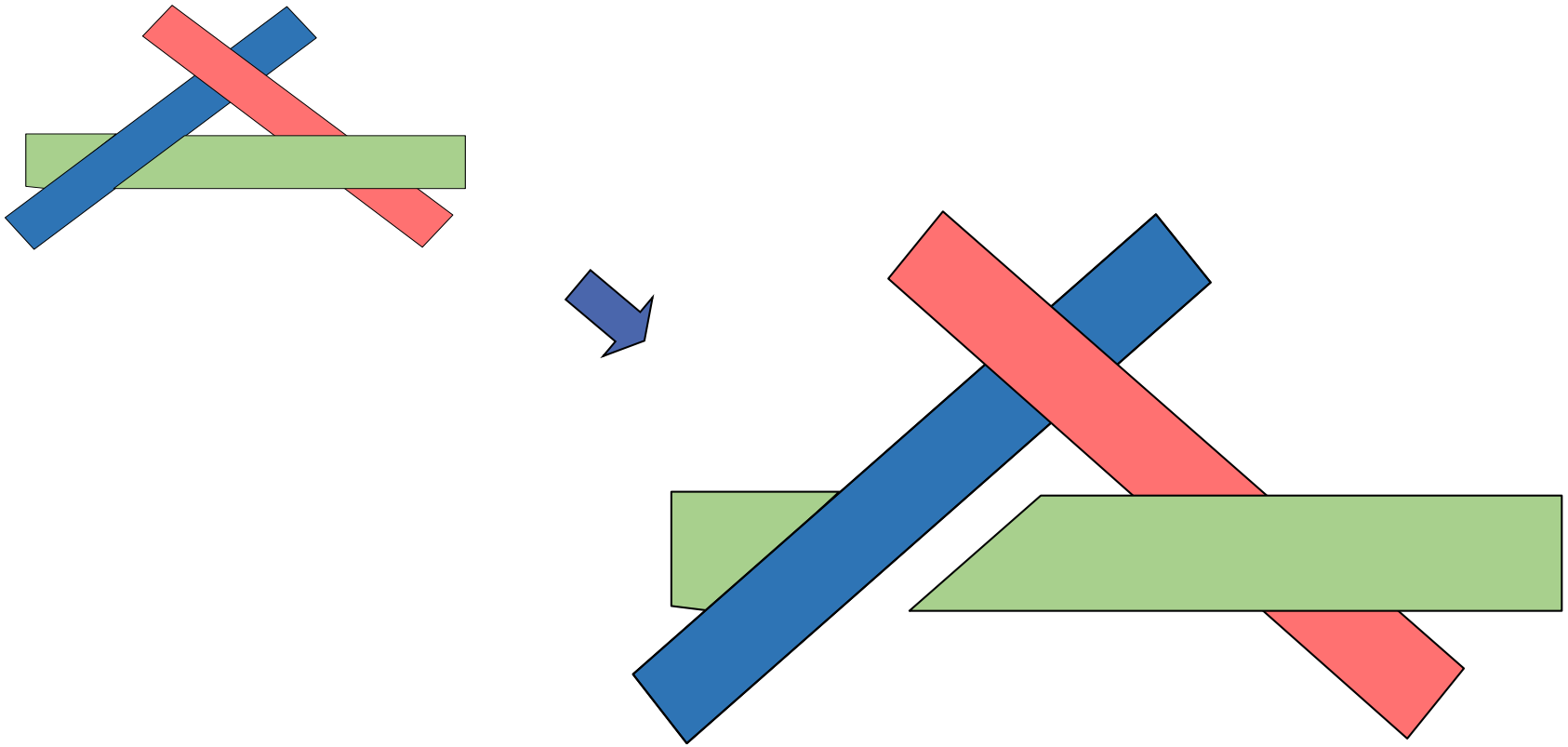
BSP STROMY


5
6
1
b
a
2
3
b
a
4
7

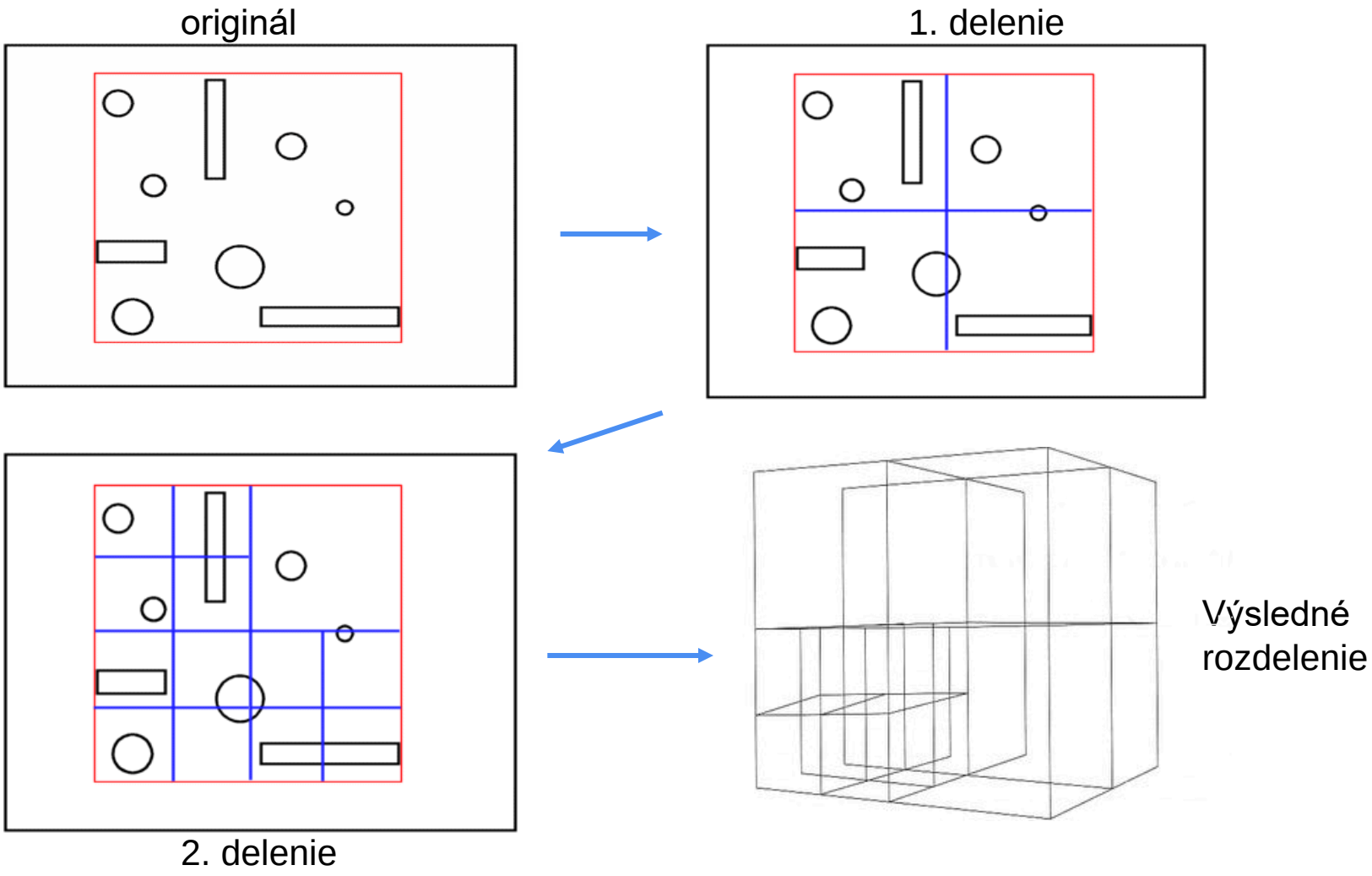

2,7
5
3a
6
1a
1b
4
3b


IV
5
V
VIII
VII
6
3
b
b
4
VI
IX
a
2
a
7
1
II
III
I

BSP STROMY – RIEŠENIE CYKLICKEJ SCÉNY

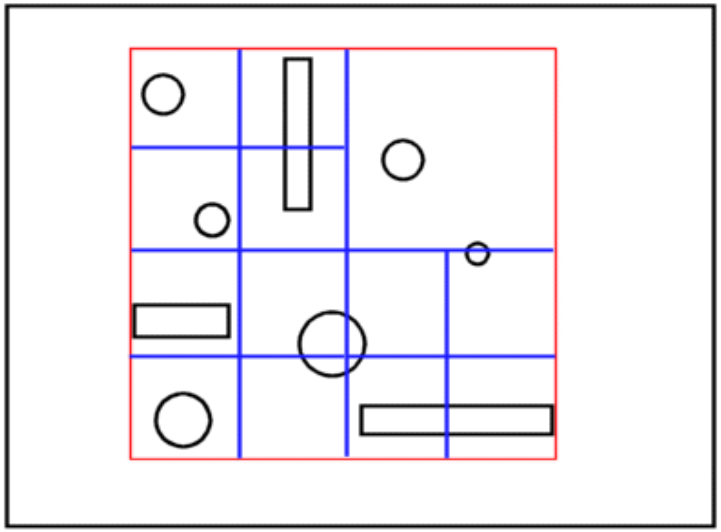


OKTANTOVÉ STROMY (OCTREES)

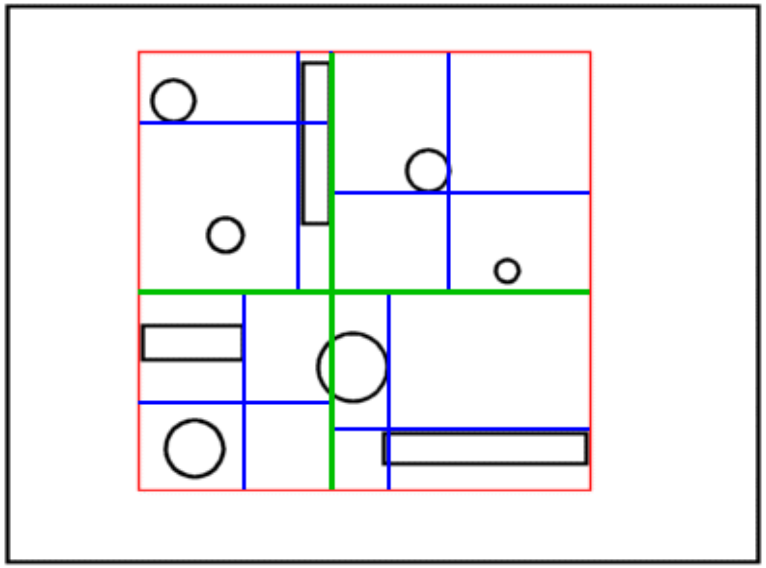


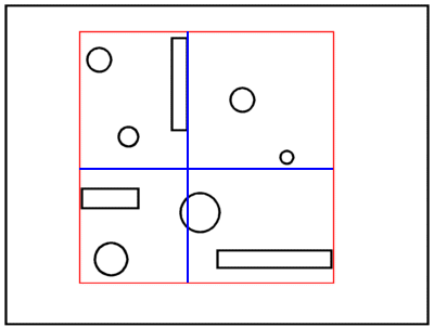
OKTANTOVÉ STROMY

Pôvodné rozdelenie



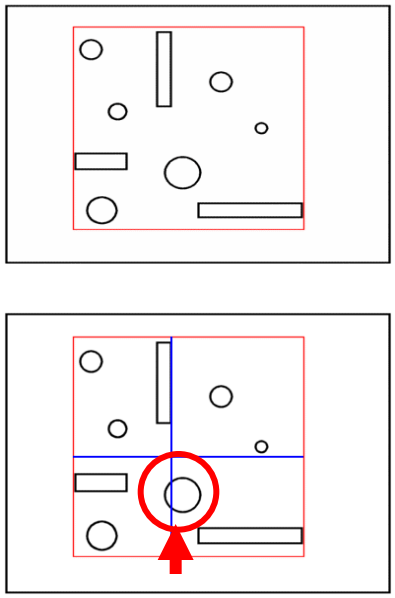
Adaptívne rozdelenie



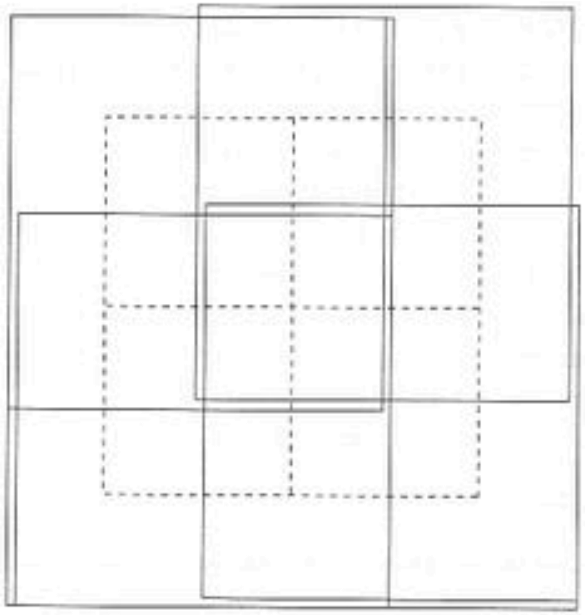


OKTANTOVÉ STROMY

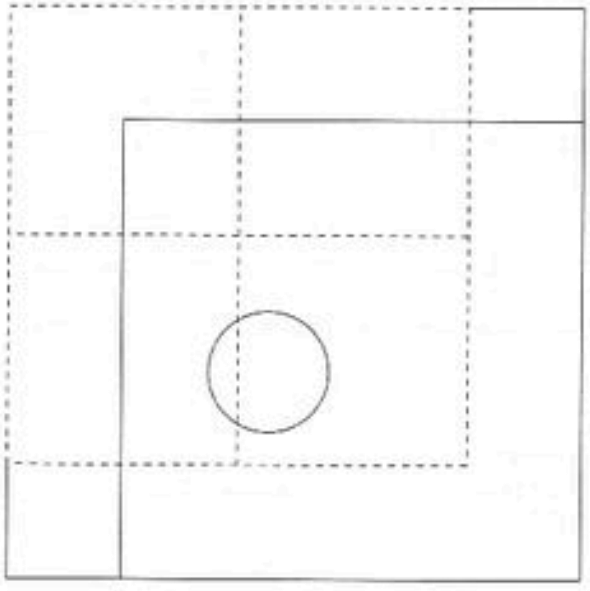
originál



Delenie pri použití adaptívneho delenia

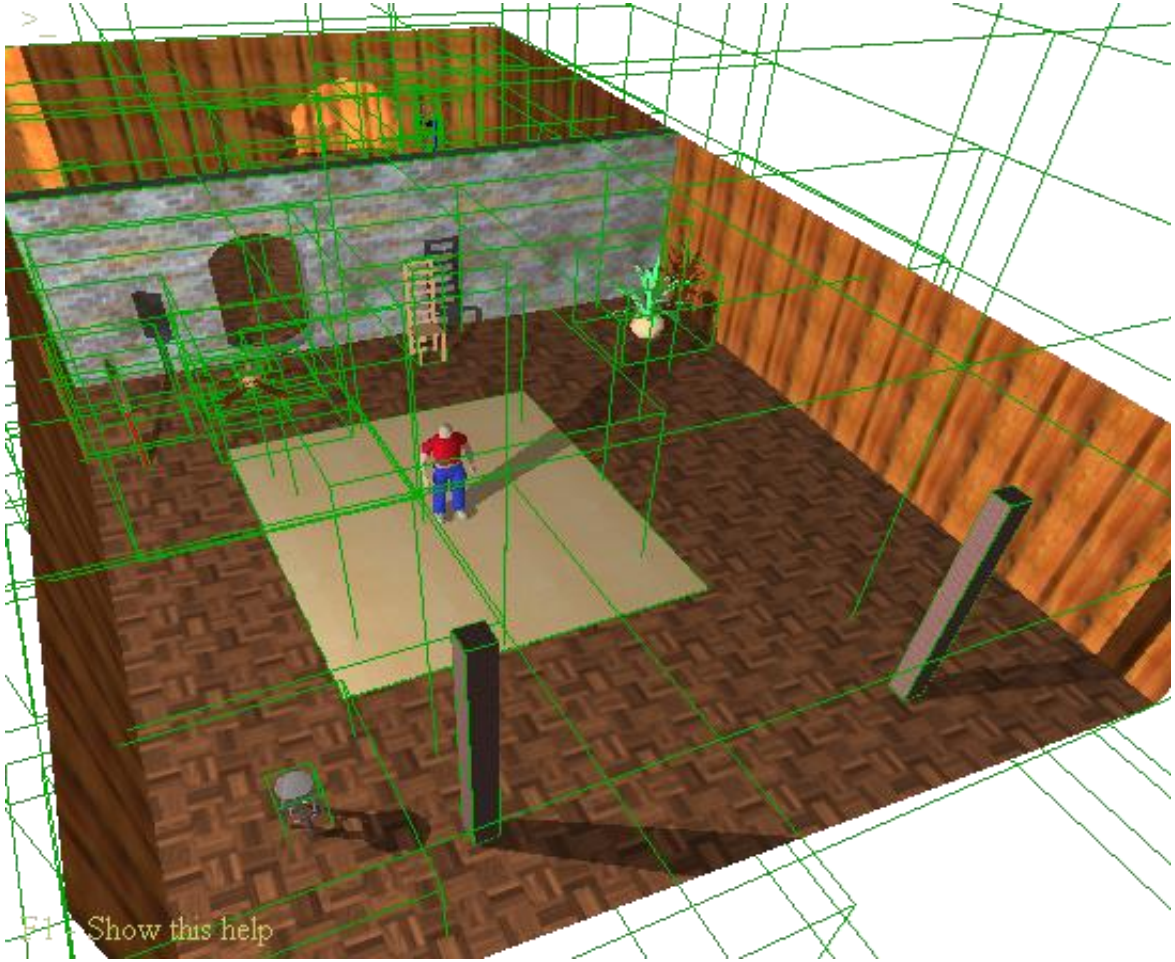


Stratové delenie



Umiestnenie problematického objektu v stratovom oktantovom strome (pravý dolný uzol)

OKTANTOVÉ STROMY

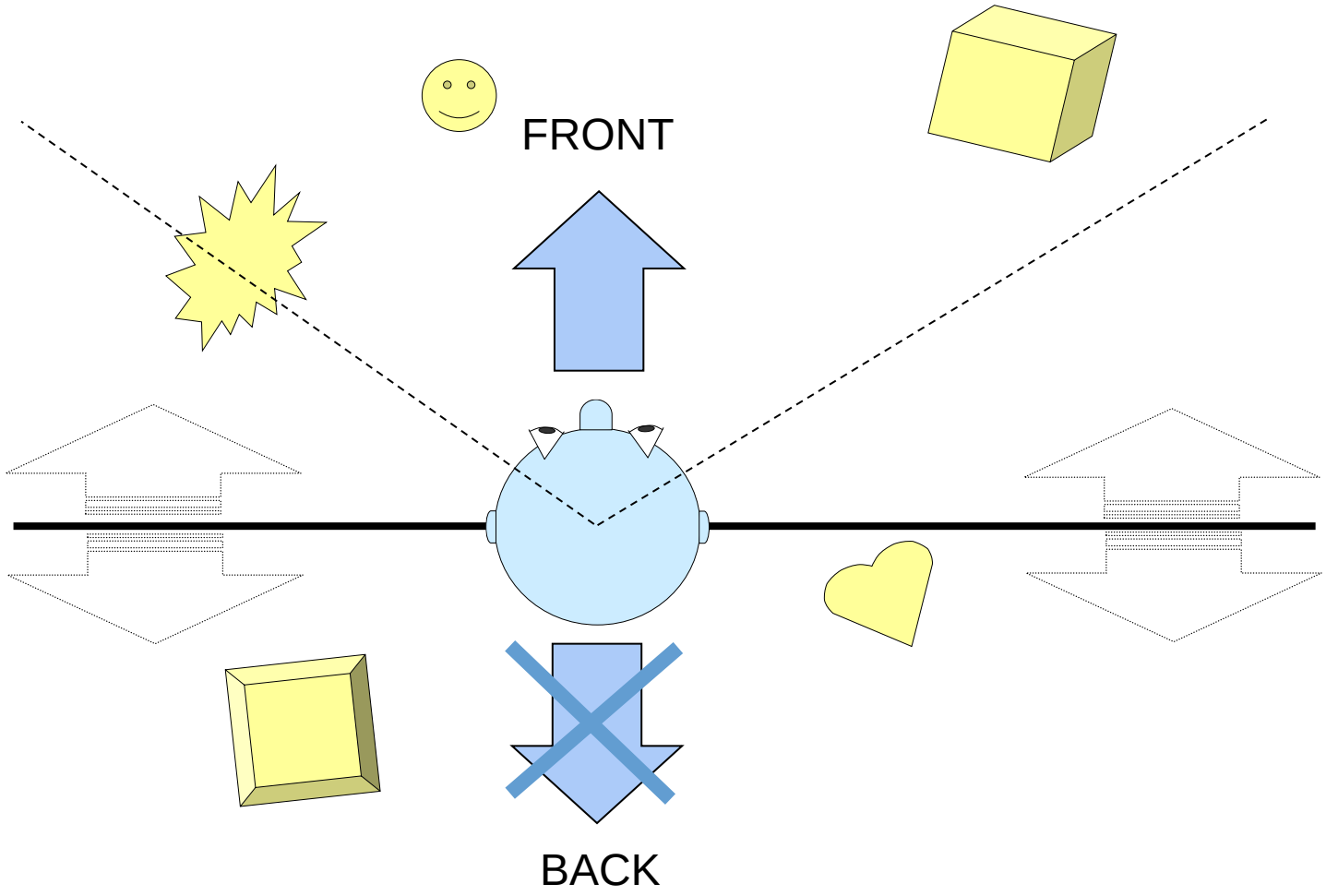


Príklad zložitejšieho delenia priestoru pomocou oktantových stromov

Urýchľovacie metódy riešenia viditeľnosti

- FV (Front view)/ BC (Back Cut)
- Orezávanie na zorný ihlan
- Ohraničujúce objekty
- Sektorovanie
- Potenciál viditeľnosti (PVS)
- S-buffer

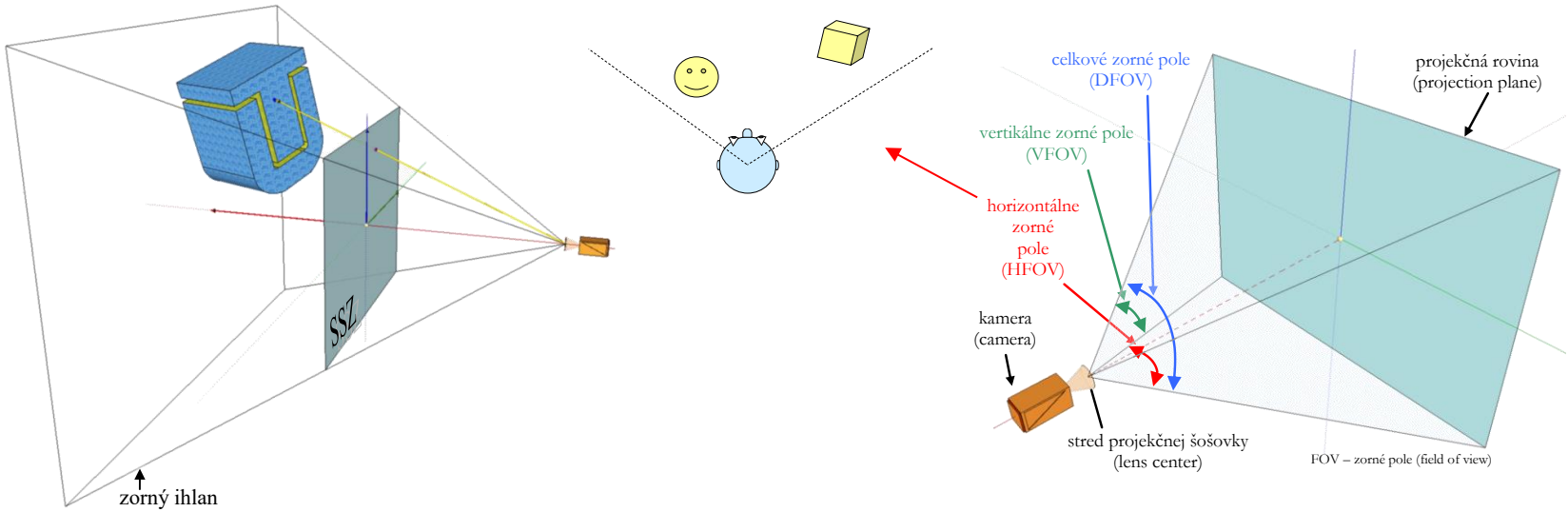
FRONT VIEW / BACK CUT



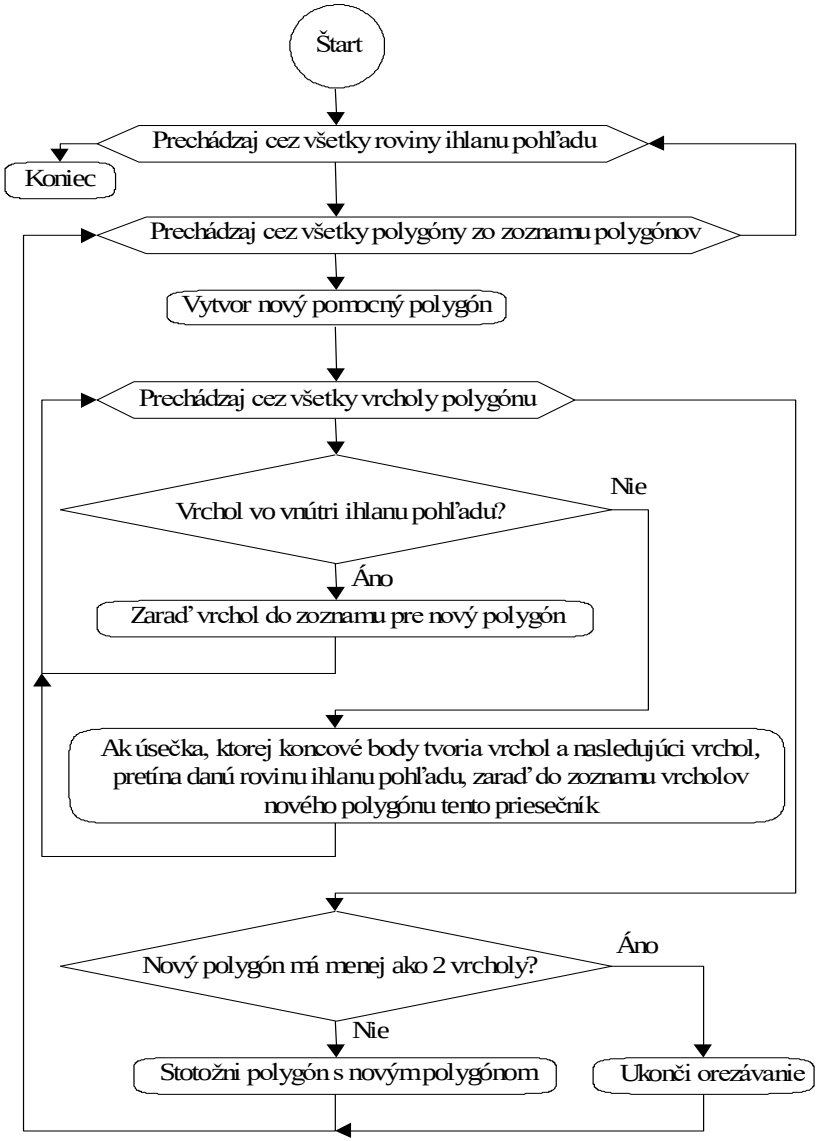
OREZÁVANIE NA ZORNÝ IHLAN

Zorný ihlan je časť priestoru, ktorá je vymedzená *H/V-FOV* a po projekčnej a zobrazovacej transformácii (*USS*→ *SSC*→*SSZ*) zobrazí do obdĺžnika na priemetni, ktorý reprezentuje tú jej časť, ktorá v konečnom dôsledku bude viditeľná na zobrazovači (obrazovke) t.j. zorný priestor.

Orezávanie je možné robiť buď v *SSC* alebo v *USS*.

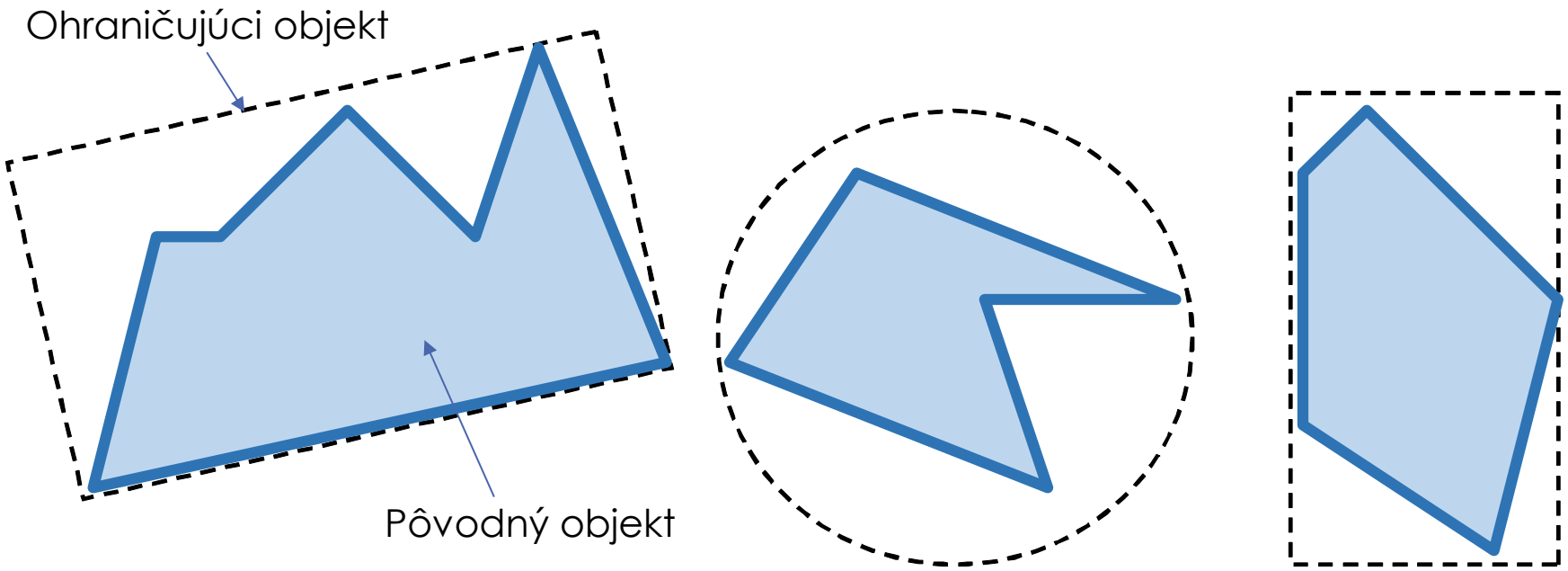


OREZÁVANIE NA ZORNÝ IHLAN - ALGORITMUS



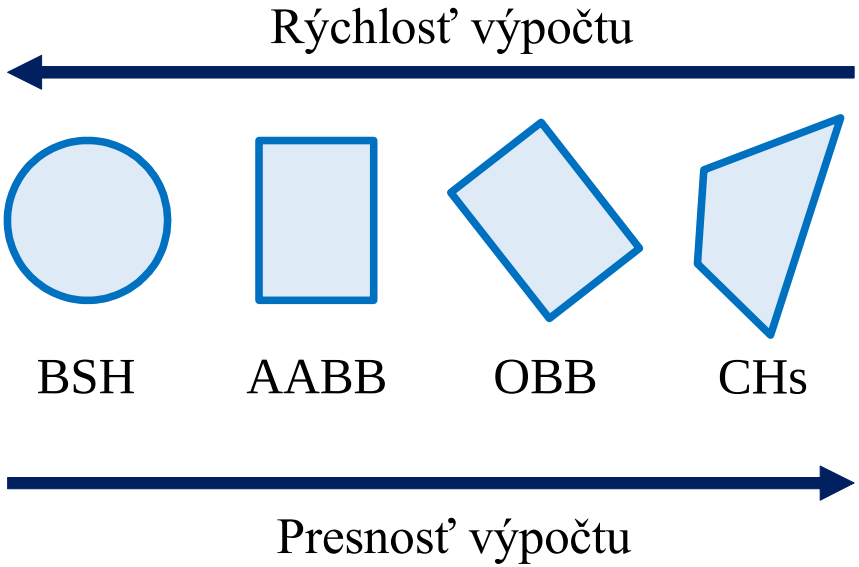
OHRANIČUJÚCE OBJEKTY

Pri metóde ohraničujúcich objektov sa zrýchlenie dosahuje výpočtom nad geometricky jednoduchším ohraničujúcim objektom oproti zložitejšiemu pôvodnému objektu. Ak je viditeľný ohraničujúci objekt, je viditeľný aj pôvodný ohraničený objekt. Táto metóda sa používa aj v prípade detekcie kolízií.



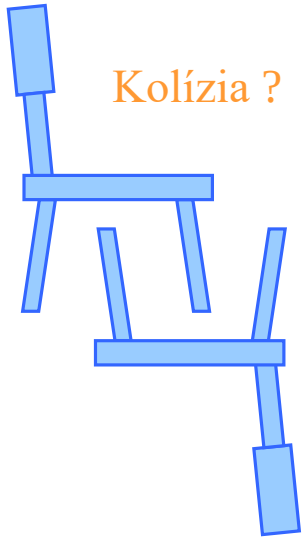
OHRANIČUJÚCE OBJEKTY

- Hierarchia ohraničujúcich gulí (BSH)
- Osovo - orientovaný kváder (AABB)
- Objektovo - orientovaný kváder (OBB)
- Konvexná obálka (Convex Hull)



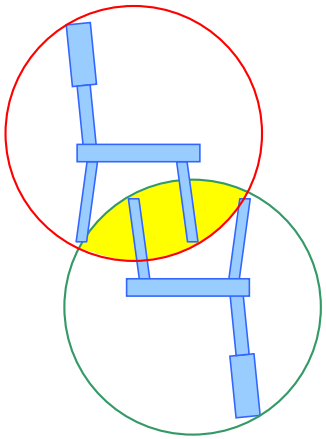
OHRANIČUJÚCE OBJEKTY – KOLÍZIE

PRÍKLAD RIEŠENIA KOLÍZIE POMOCOU HIERARCHIE GULÍ V PROJEKCII 2D



Scéna pre detekciu kolízie

1

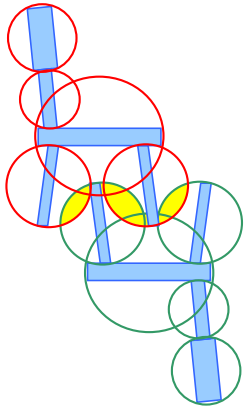


Detekcia kolízie na 1. stupni hierarchie

Kolízia ?

Áno

2

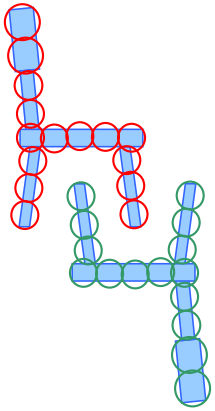


Detekcia kolízie na 2. stupni hierarchie

Kolízia ?

Áno

3



Detekcia kolízie na 3. stupni hierarchie

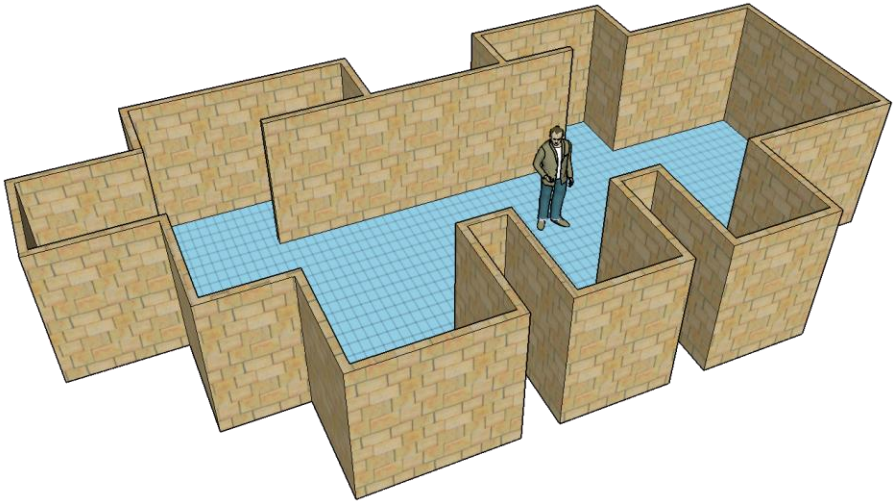
Kolízia ?

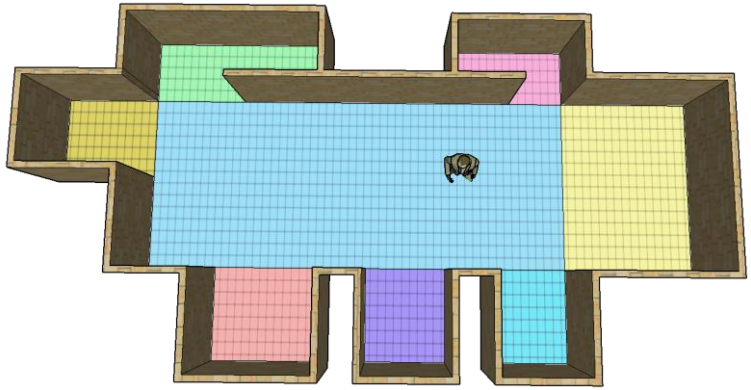
Nie

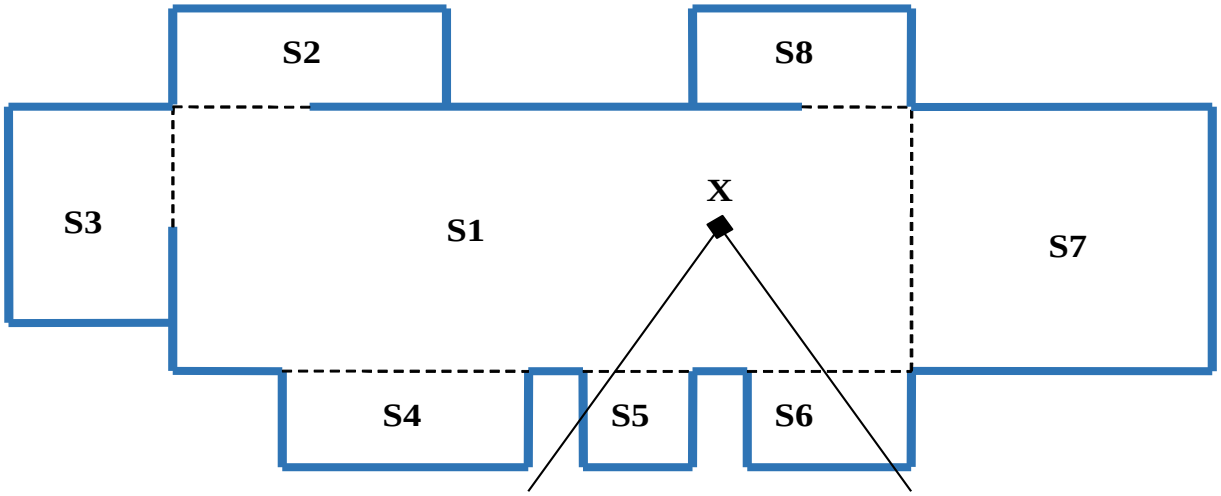
Kolízia ?
(výsledok)

Nie

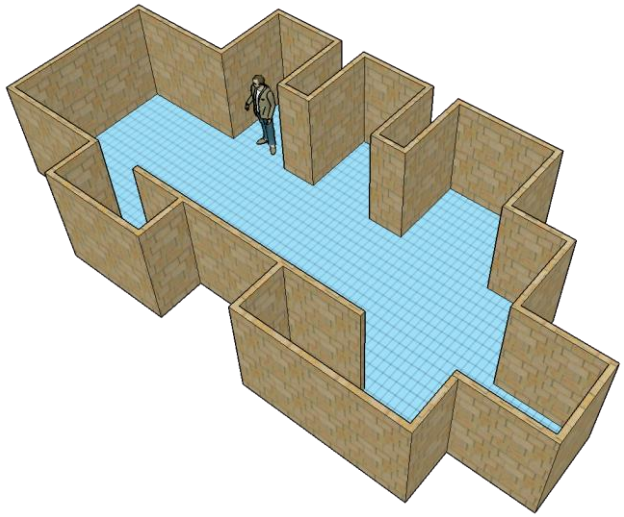
SEKTOROVANIE

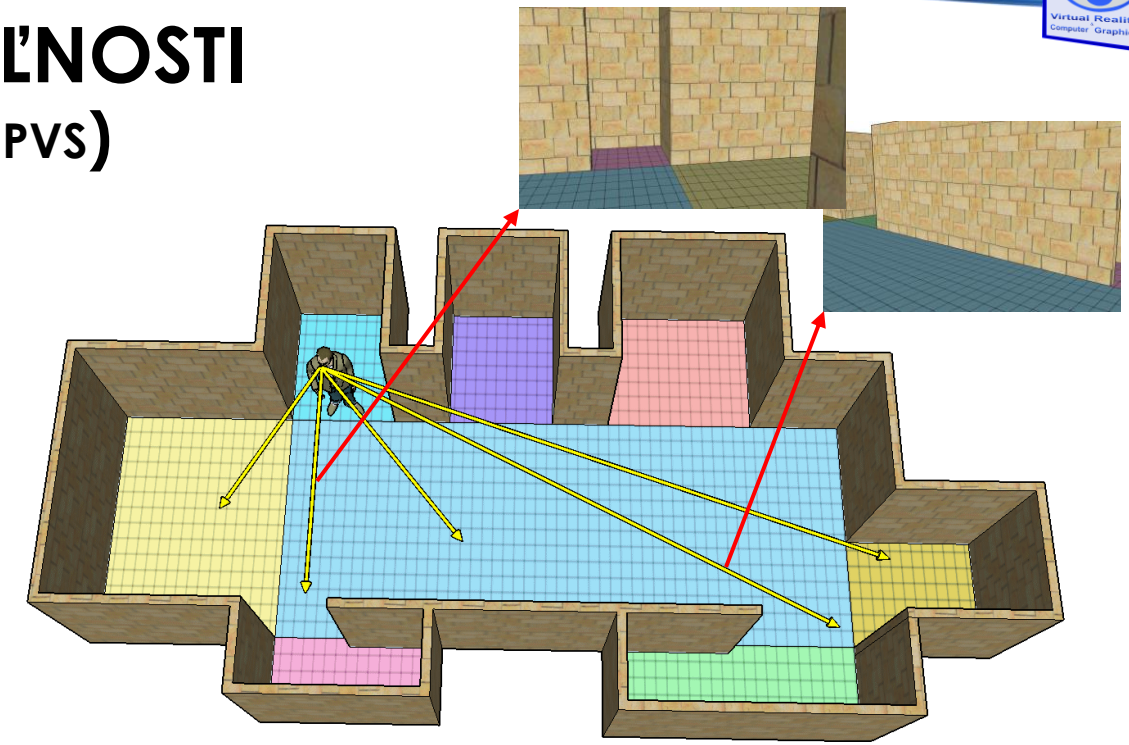


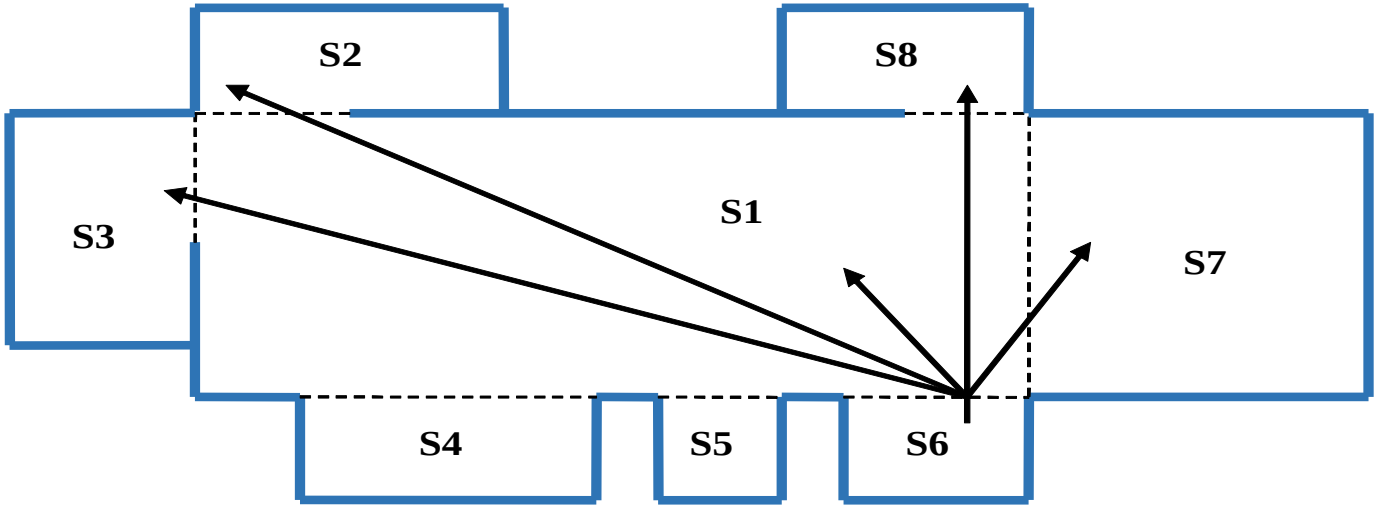




POTENCIÁL VIDITEĽNOSTI
(POTENTIALLY VISIBLE SET, PVS)







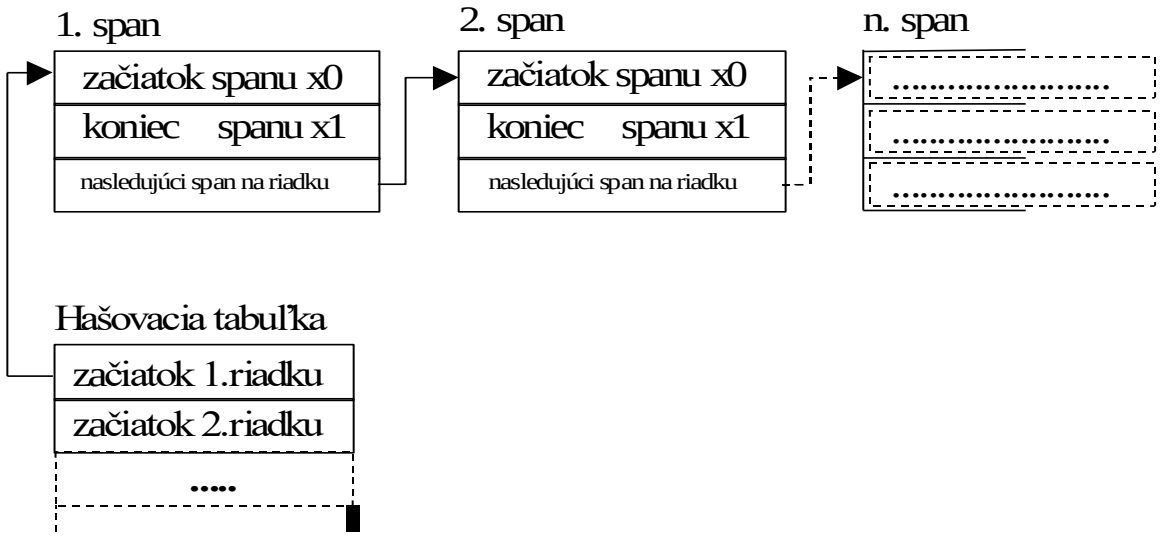
S-BUFFER (SPAN BUFFER)

je jednou z techník, ako zabezpečiť neprekresľovanie už vykreslených polygónov v rámci SSZ, jedná sa teda o algoritmus, ktorý pracuje prioritne v 2D.

Jeho použitie sa teda obmedzuje na použitie v systémoch, ktoré kreslia scénu spredu dozadu.

S-BUFFER – ÚDAJOVÉ ŠTRUKTÚRY

Údajové štruktúry, ktoré sa používajú pri S-buffri sú rôzne. Často sa však používa najmä spojkový zoznam. Ide teda o spojkový zoznam častí riadkov (angl. span), ktoré sú už vykreslené. Takýto zoznam je zostavený osobitne pre každý riadok obrazovky a smerníky na prvý span na každom riadku sú uložené v hašovacej tabuľke.



S-BUFFER - SPRACOVANIE

Pri kreslení polygónu (mnohouholníka) sa musí každý *span* podrobiť testu, či nie je prekrytý iným, už nakresleným spanom. Ak je prekrytý, t.j. leží celý v rozmedzí x_0 a x_1 nejakého spanu v danom riadku, tak sa nekreslí. V opačnom prípade je potrebné počítať prieniky spanov a vykresliť len tie časti, ktoré sú ešte nevykreslené, t.j. postupne zaplňovať diery v S-buffri. Rutina na vkladanie spanu nie je vôbec jednoduchá a v značnej miere na jej efektívnosti závisí aj výkon vizualizačného systému.

Q & A

branislav.sobota@tuke.sk

Katedra počítačov a informatiky, FEI TU v Košiciach