

Základy počítačové grafiky

Řešení viditelnosti objektů ve 3D

Michal Španěl
Tomáš Milet

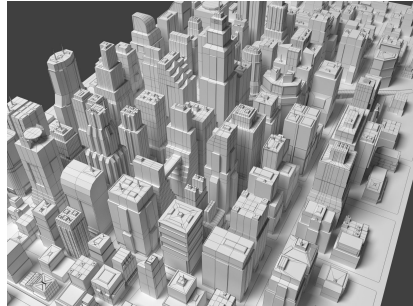


Ústav počítačové grafiky a multimédií

Brno 2021

Cíl přednášky

Seznámit se s principy nalezení těch částí objektů, které jsou viditelné z místa pozorovatele



Obsah

1 Úvod

- Principy a akcelerační struktury
- Algoritmy viditelnosti

2 Viditelnost hran

- Robertsův algoritmus

3 Objektové algoritmy

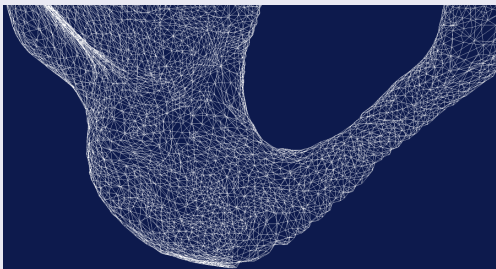
- Malířův algoritmus
- Dělení obrazu

4 Rastrové algoritmy

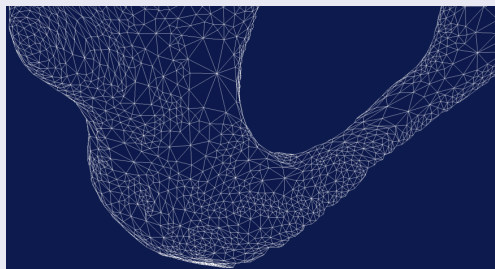
- Z-buffer
- Kreslení s průhledností
- Ray-casting

Proč řešíme viditelnost?

Drátový model bez řešení viditelnosti

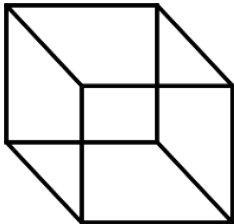


Drátový model s řešením viditelnosti

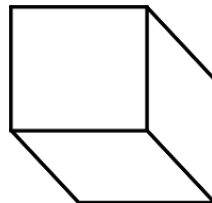
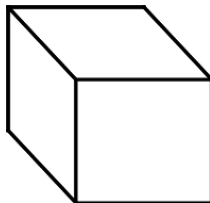


Proč řešíme viditelnost?

- Drátový model kostky

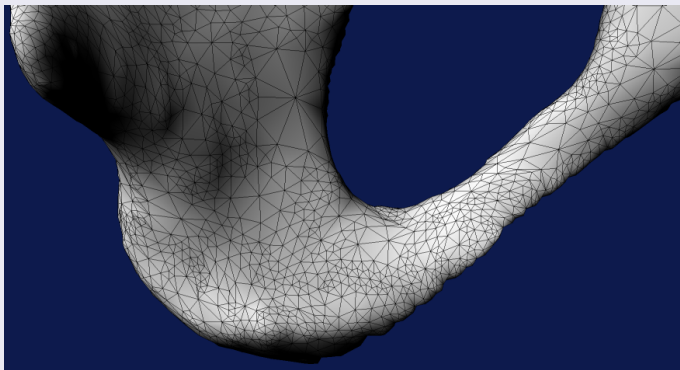


- ?

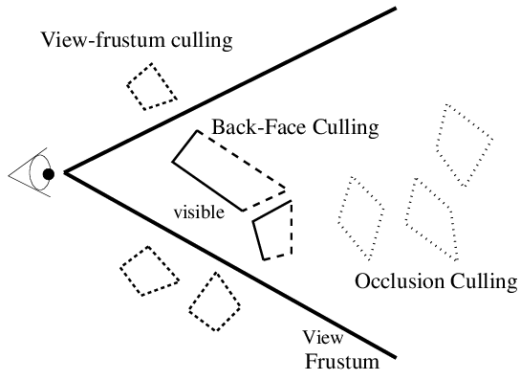


Proč řešíme viditelnost?

Model s řešením viditelnosti



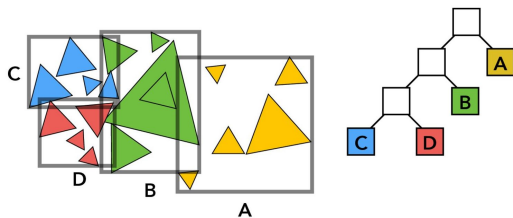
Viditelnost se řeší na několika úrovních...



- View-Frustum culling - tj. ořezání pohledovým jehlanem
- Back-Face culling - zadní stranu nevidíme
- Occlusion culling - co je zakryto jiným objektem nevidíme (tzv. occluder)



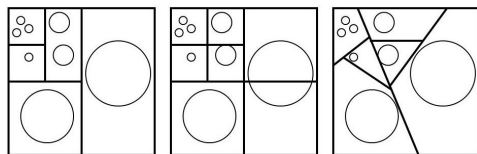
View-Frustum Culling



- Nelze testovat každý trojúhelník...

Metody dělení prostoru

- Obalová tělesa (např. BVH - Bounding Volume Hierarchies)
- Stromové struktury (BSP tree, octree, kd-tree, atd.)

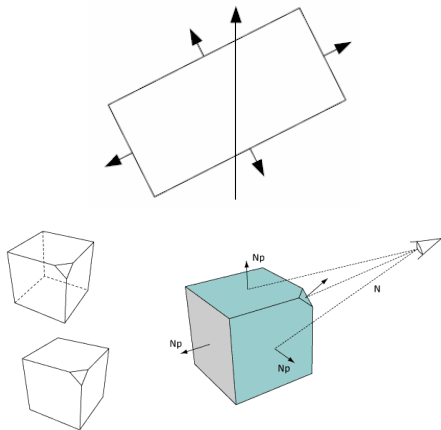


KD tree

Quadtree (2D)
Octree (3D)

BSP tree

Back-Face Culling



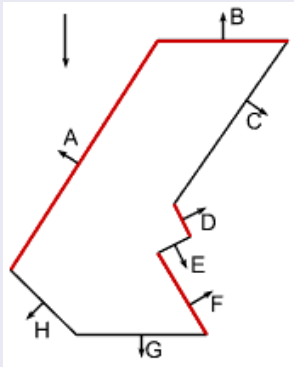
- Řeší viditelnost ploch (*Hidden Surface Removal*)
- Viditelné plochy jsou přivrácené k pozorovateli – **mají normálu směrem k pozorovateli**

Test orientace normály vůči vektoru pohledu

- Skalární součin.

Motivace pro pokročilejší algoritmy

Nekonvexní tělesa (+ protínání obrysových hran)



Překrývání více objektů ve scéně



Algoritmy viditelnosti

Vektorový výsledek

- Výsledkem je soubor viditelných a neviditelných ploch, hran, apod.

Rastrový výsledek

- Obraz viditelných ploch, barva osvětlení, stínování, apod.

Objektové

- Pro každý objekt scény hledá viditelné části – složitost n^2

Obrazové

- Pro každý pixel obrazu hledá viditelný objekt – složitost $n * p$

- V praxi kombinace přístupů - např. tiled-rendering

Algoritmy viditelnosti

Vektorový výsledek

- Výsledkem je soubor viditelných a neviditelných ploch, hran, apod.

Rastrový výsledek

- Obraz viditelných ploch, barva osvětlení, stínování, apod.

Objektové

- Pro každý objekt scény hledá viditelné části – složitost n^2

Obrazové

- Pro každý pixel obrazu hledá viditelný objekt – složitost $n * p$

- V praxi kombinace přístupů - např. tiled-rendering

Algoritmy viditelnosti

Vektorový výsledek

- Výsledkem je soubor viditelných a neviditelných ploch, hran, apod.

Rastrový výsledek

- Obraz viditelných ploch, barva osvětlení, stínování, apod.

Objektové

- Pro každý objekt scény hledá viditelné části – složitost n^2

Obrazové

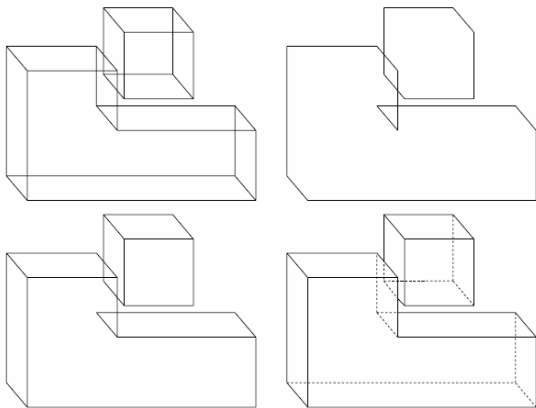
- Pro každý pixel obrazu hledá viditelný objekt – složitost $n * p$

- V praxi kombinace přístupů - např. tiled-rendering

Obsah

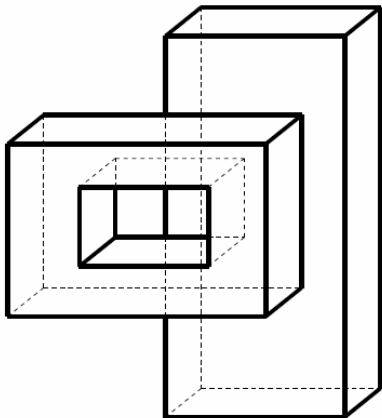
- 1 Úvod
 - Principy a akcelerační struktury
 - Algoritmy viditelnosti
- 2 Viditelnost hran
 - Robertsův algoritmus
- 3 Objektové algoritmy
 - Malířův algoritmus
 - Dělení obrazu
- 4 Rastrové algoritmy
 - Z-buffer
 - Kreslení s průhledností
 - Ray-casting

Viditelnost hran (angl. hidden line removal)



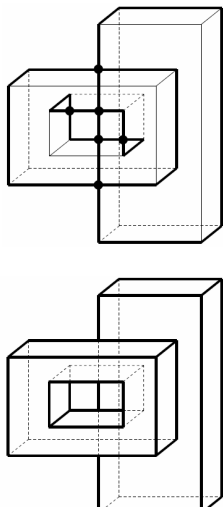
- Hrana mezi viditelnými plochami je *potenciálně viditelná*.
- Hrana mezi neviditelnými plochami je *neviditelná*.
- Hrana mezi viditelnou a neviditelnou plochou je *obrysová*.

Robertsův algoritmus



- Klasický vektorový algoritmus.
- Dělení potenciálně viditelných hran na úseky, kde se mění viditelnost.
- Vykreslení viditelných úseků.

Robertsův algoritmus, pokr.



Postup

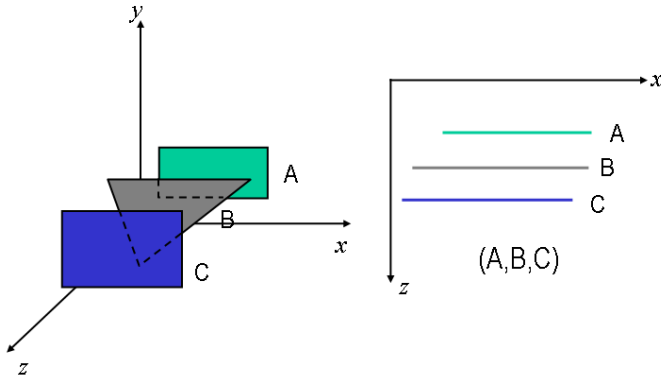
- Získat viditelné a obrysové hrany.
- Rozdělit na *úseky konstantní viditelnosti* (průsečíky s obrysovými hranami).
- Testovat viditelnost úseků, podle vzdálenosti průsečíku a zakrytí.
 - *Viditelnost se mění na obrysových hranách.*
 - *Střídají se úseky viditelné a zakryté.*

Obsah

- 1 Úvod
 - Principy a akcelerační struktury
 - Algoritmy viditelnosti
- 2 Viditelnost hran
 - Robertsův algoritmus
- 3 Objektové algoritmy
 - Malířův algoritmus
 - Dělení obrazu
- 4 Rastrové algoritmy
 - Z-buffer
 - Kreslení s průhledností
 - Ray-casting

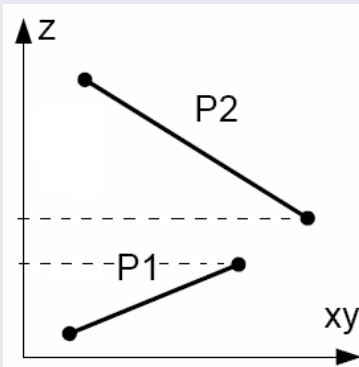
Malířův algoritmus (Painter's Algorithm)

- "Objektový" algoritmus řešení viditelnosti – vykreslování odzadu dopředu
- Seřazení polygonů podle vzdálenosti (*depth sorting algorithm*)

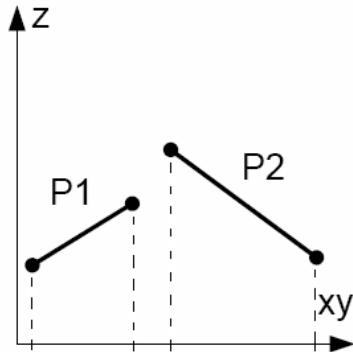


Testy překrývání ploch

$$z1_{max} < z2_{min}$$

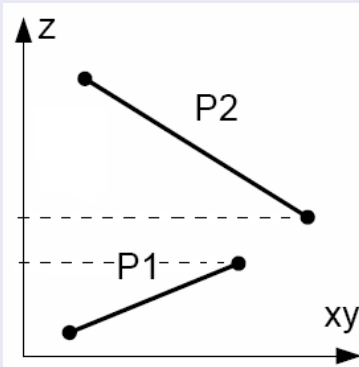


Nepřekrývají se průměty ploch v rovině XY

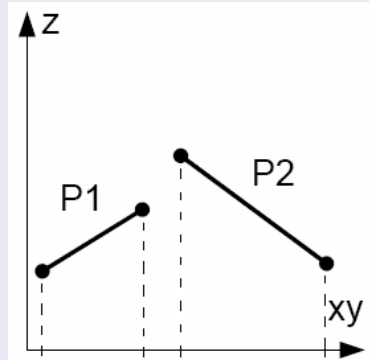


Testy překrývání ploch

$$z1_{max} < z2_{min}$$

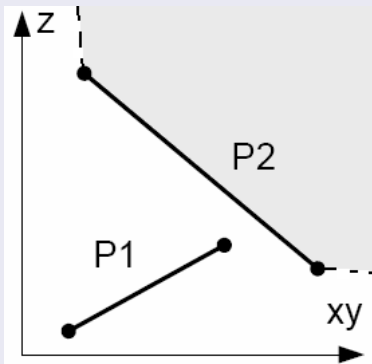


Nepřekrývají se průměty ploch v rovině XY

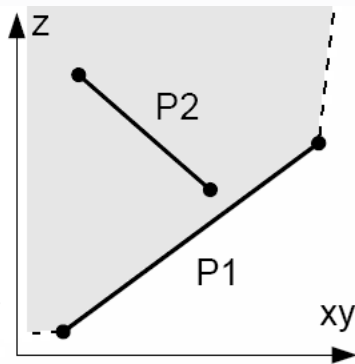


Testy překrývání ploch, pokr.

Plocha P1 leží v přivráceném poloprostoru plochy P2

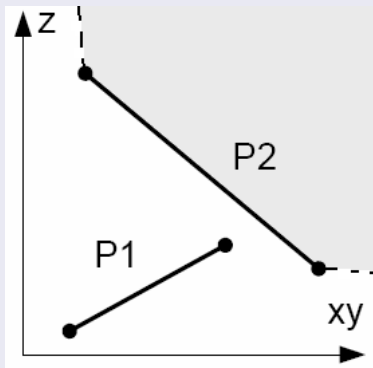


Plocha P2 leží v odvráceném poloprostoru plochy P1

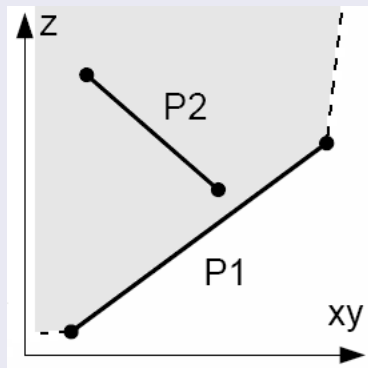


Testy překrývání ploch, pokr.

Plocha P1 leží v přivráceném poloprostoru plochy P2



Plocha P2 leží v odvráceném poloprostoru plochy P1



Malířův algoritmus, pokr.

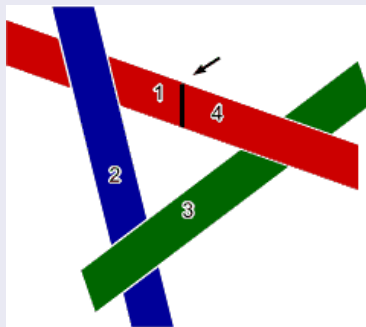
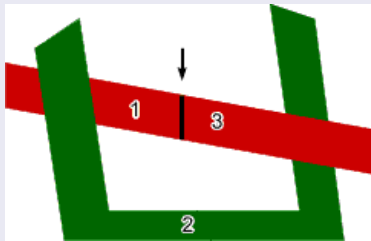
Postup

- Seřazení ploch podle z_{min} .
- Testování aktuální nejvzdálenější plochy P_2 na překrytí "bližšími":
 - Při nesplnění některého testu → prohození porovnávané plochy s aktuální v seznamu.
- Vykreslení a vyřazení aktuální plochy nepřekrývá-li jinou.
- Opakování testování.

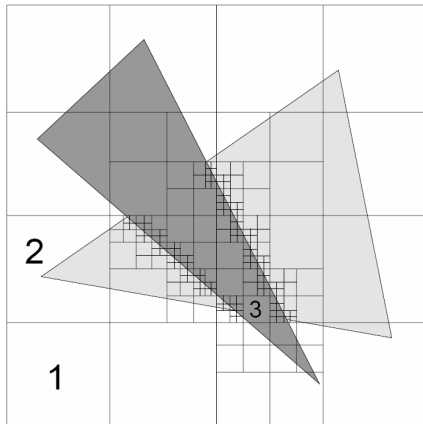
Malířův algoritmus, pokr.

Problémy se vzájemně se překrývajícími objekty

- Dělení na části.



Dělení obrazu



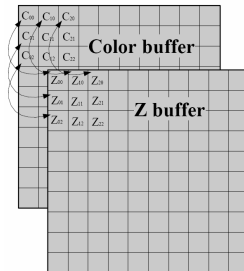
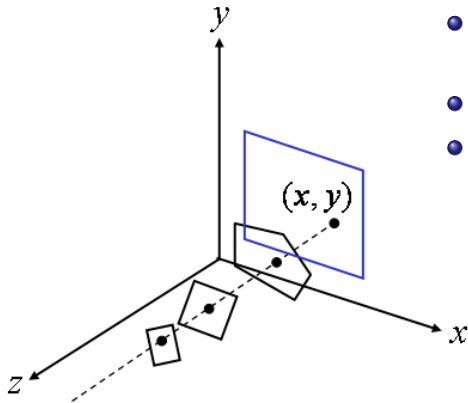
- Warnock J., 1969
- Rastrový objektový algoritmus.
- Dělení okna pomocí quad-tree, dokud není vyplněno jedním objektem
 - Žádný objekt v okně – pozadí.
 - Jeden objekt v okně – vykreslit.
 - Více objektů v okně, ale nejblíže překrývá ostatní – vykreslit.
 - Jinak → dělení okna.
- Při velikosti okna 1 pixel konec.

Obsah

- 1 Úvod
 - Principy a akcelerační struktury
 - Algoritmy viditelnosti
- 2 Viditelnost hran
 - Robertsův algoritmus
- 3 Objektové algoritmy
 - Malířův algoritmus
 - Dělení obrazu
- 4 Rastrové algoritmy
 - Z-buffer
 - Kreslení s průhledností
 - Ray-casting

Z-buffer (depth buffer)

- Rastrový obrazový algoritmus
- Paměť hloubky obsahuje Z souřadnice nejbližších bodů ploch
- Plochy není třeba třídit!
- Rychlý algoritmus, snadná realizace v HW

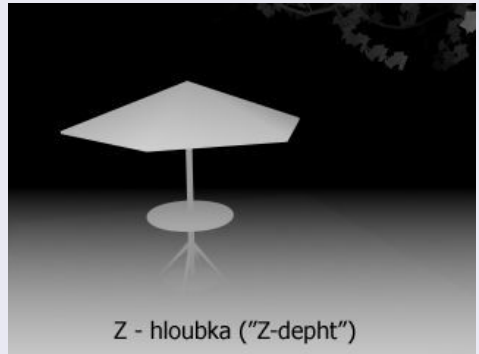


Příklad – Z-buffer

Obsah color bufferu

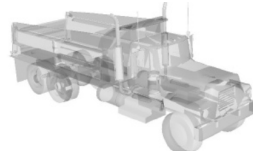


Obsah Z-bufferu

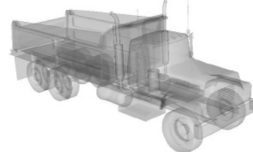
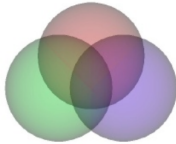


Kreslení průhledných objektů (Order-Independent Transparency)

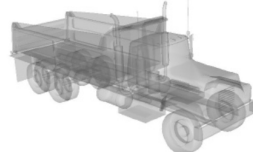
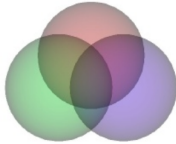
OpenGL



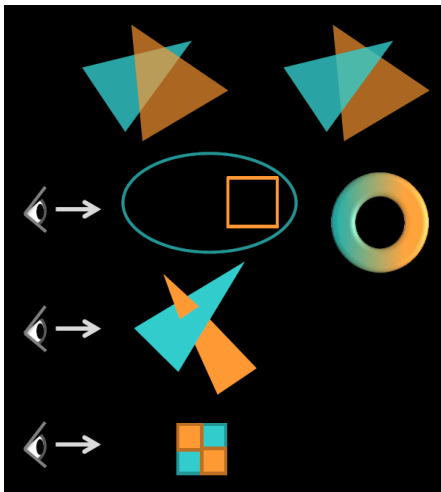
Dual Depth
Peeling



Weighted
Average



OIT – Problémy



- Záleží na pořadí – alpha blending není komutativní
- Třídění na úrovni objektů nestačí...
- Třídění troj. je náročné... a nestačí...
- Ideálně potřebujeme třídit fragmenty

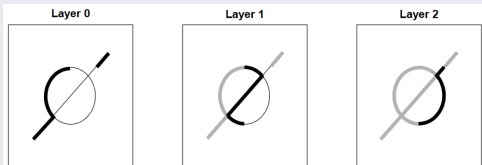
OIT – Možná řešení

Chytřejší rovnice pro blending

- Pro jednoduché vizualizace stačí
- Weighted average, ...

Multi-pass rendering

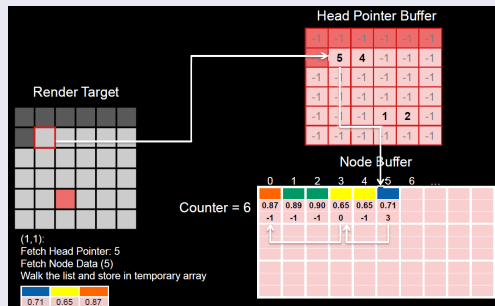
- Více průchodů scénou na frame
- Depth Peeling, ...



(FIT VUT v Brně)

Třídění fragmentů

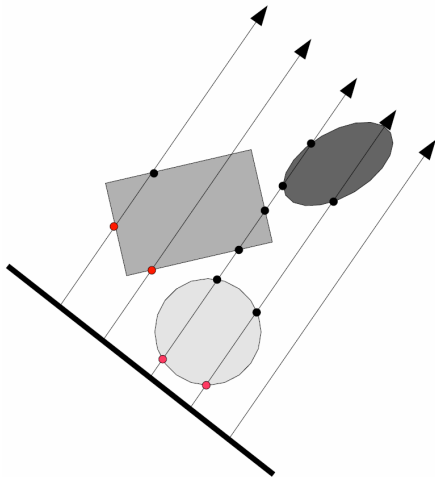
- Např. Linked-lists



Základy počítačové grafiky

28 / 32

Ray-casting



- Rastrový obrazový algoritmus řešení viditelnosti.
- **Vrhání paprsků z místa pozorovatele (rovnoběžně, perspektivně).**
- Průsečíky paprsků se všemi objekty podél jejich dráhy, výběr nejbližšího.
- Pomalejší než Z-buffer, lze použít i pro průhledné objekty...

Ray-casting, pokr.

Předzpracování scény

- Vyloučení odvrácených a zakrytých částí, dělení obrazu nebo prostoru scény, ...

+/-

- Snadná realizace vizuálních efektů.
- Lze použít pro zobrazení *CSG modelů*, *implicitních ploch*, apod.
- Snadná paralelizace, ale náročnější HW realizace.

Ray-casting, pokr.

Předzpracování scény

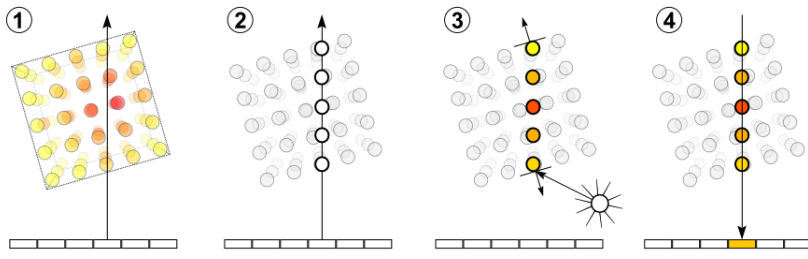
- Vyloučení odvrácených a zakrytých částí, dělení obrazu nebo prostoru scény, ...

+/-

- Snadná realizace vizuálních efektů.
- Lze použít pro zobrazení *CSG modelů*, *implicitních ploch*, apod.
- Snadná paralelizace, ale náročnější HW realizace.

Příklad – Volume Ray-casting

- Vržení paprsku
- *Vzorkování (sampling)* – interpolace hodnot voxelů
- *Stínování (shading)* – rozhoduje *gradient* (= orientace povrchu v objemu) a *poloha světla*
- Projekce výsledné hodnoty (součet, maximum, ...)



Příklad – Volume Ray-casting, pokr.

Různé výsledky

