Vylepšený koherentní hierarchický odstřel

Olivier Mattush, Jiří Bittner, Michael Wimmer

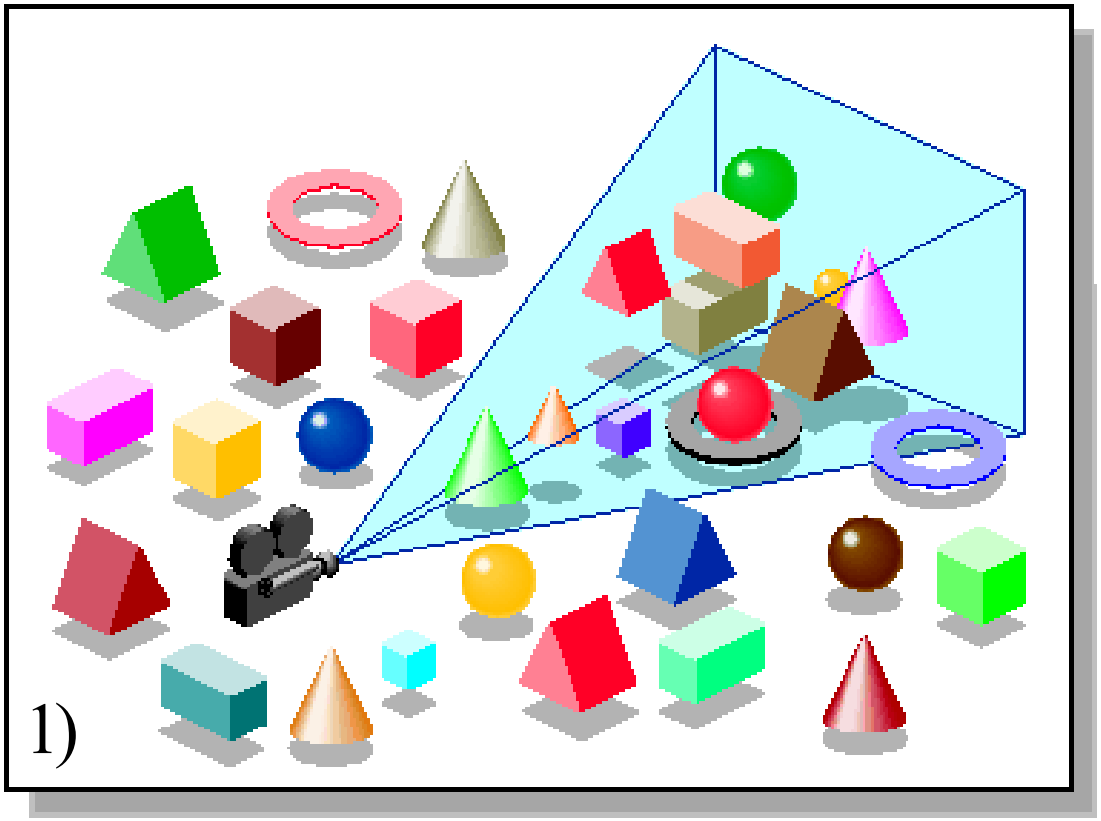
Eurographics, 2008

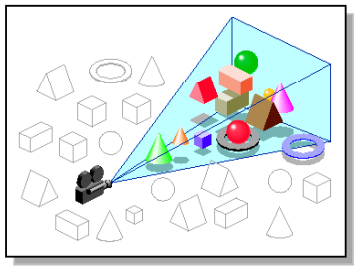
Přednáší Petr Tobola

Struktura prezentace

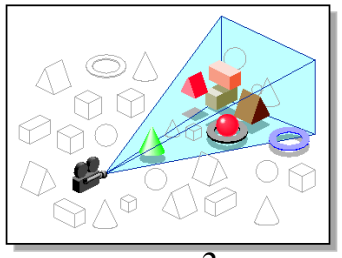
- Úvod a základní pojmy
  - Okluzní odstřel (Occlusion Culling)
  - Viditelnostní odstřel (Visibility Culling)
  - Konzervativní viditelnost
- Přístupy k řešení problému
- Původní algoritmus
  - KD-stromy
  - Hardverový Okluzní dotaz (Hardware Occlusion Query)
  - CHC
  - Optimalizace a problémy
- Vylepšení původního algoritmu
- Praktické výsledky
- Závěr

Okluzní odstřel



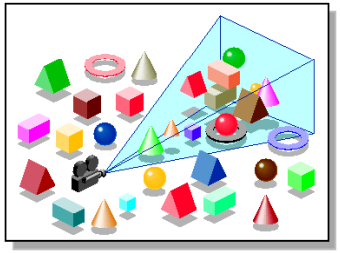


2)

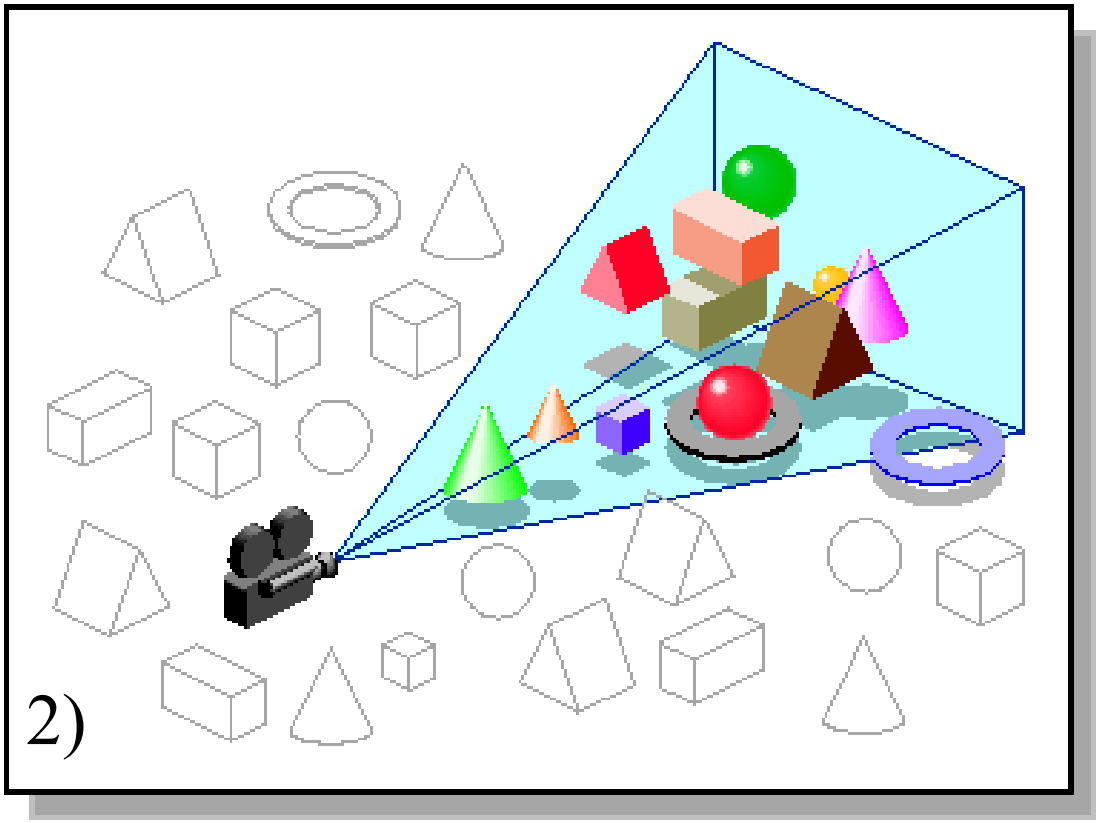


3)

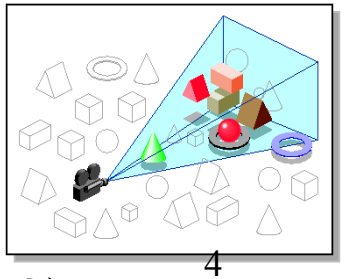
Okluzní odstřel



1)

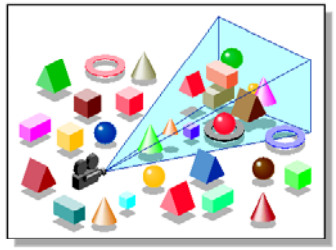


2)

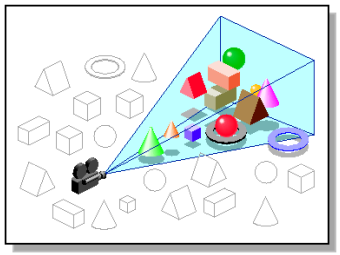


3)

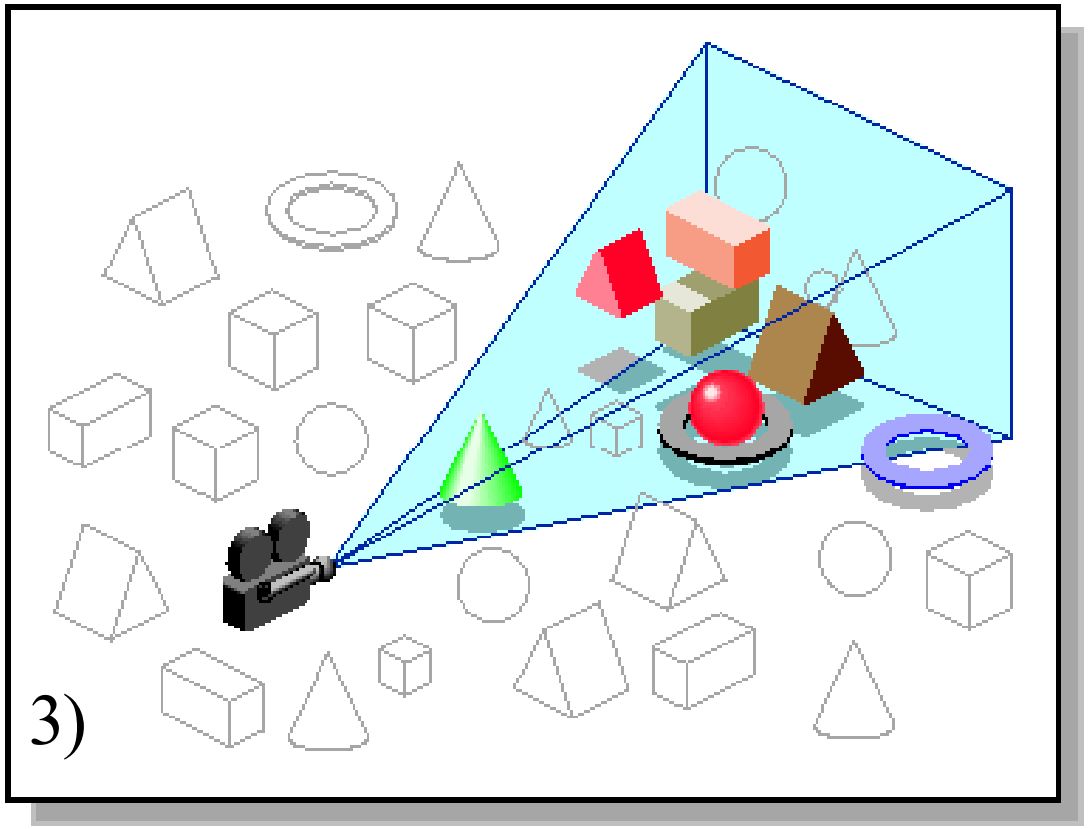
Okluzní odstřel



1)

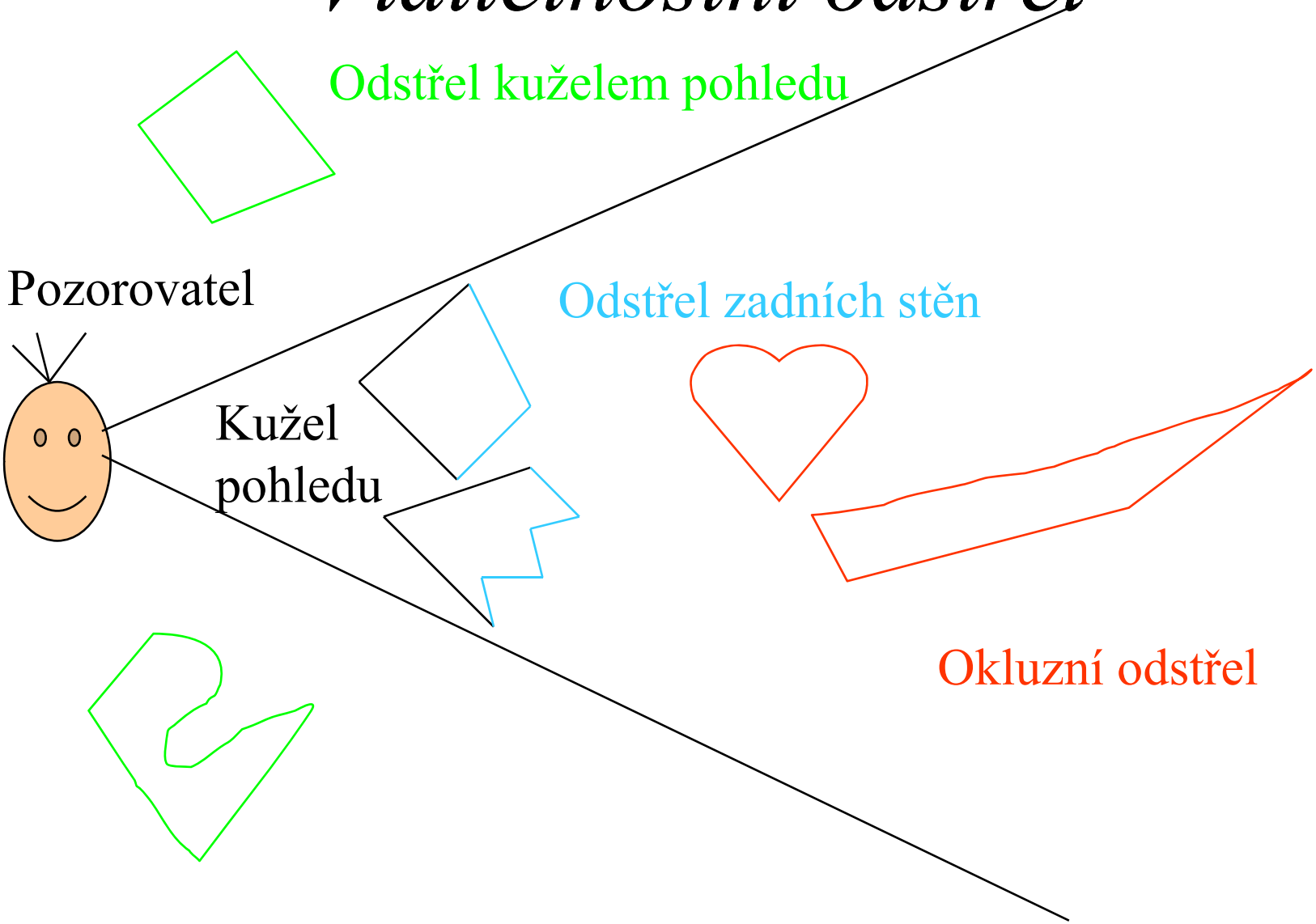


3)

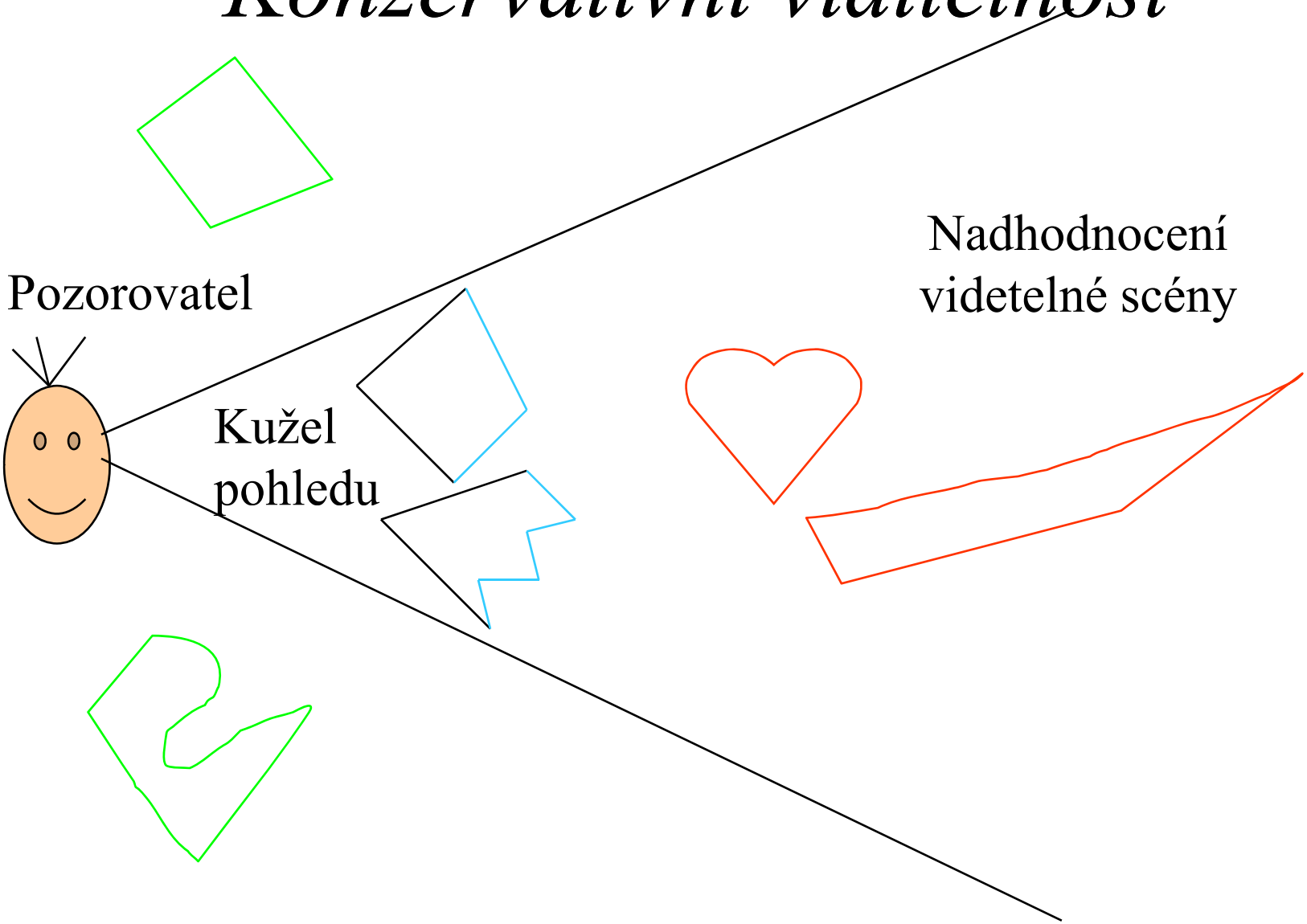


3)

Viditelnostní odstřel



Konzervativní viditelnost

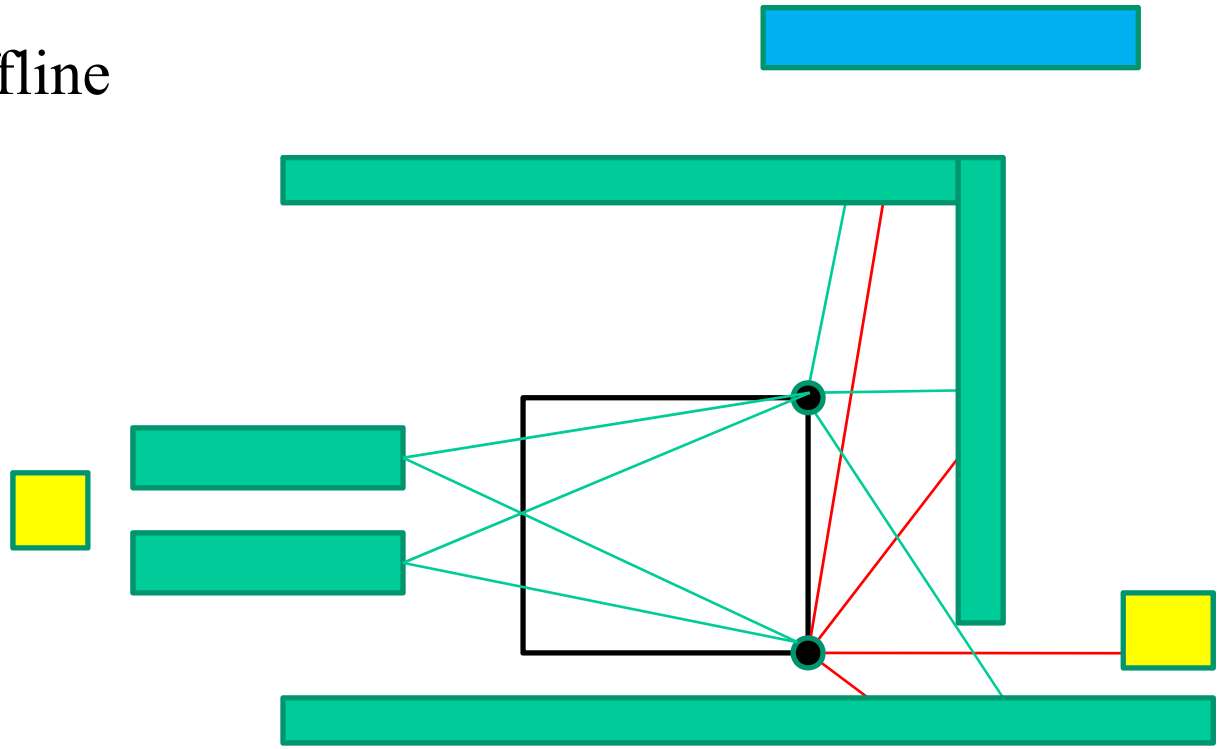


Přístupy k řešení problému

- Offline
  - Vypočti PVS (Potencially Visible Set) pro buňky pevně daného rozdělení scény
    - PVS je platná pouze pro původní pevně danou scénu
    - Pro pevně daný bod může být tato PVS nadhodnocená
    - Výpočet PVS je drahý
    - Výpočet je obtížné implementovat pro obecné scény
- Online
  - V při výpočtu každého snímku scény je potřeba udělat nějaké výpočty navíc

Přístupy k řešení problému

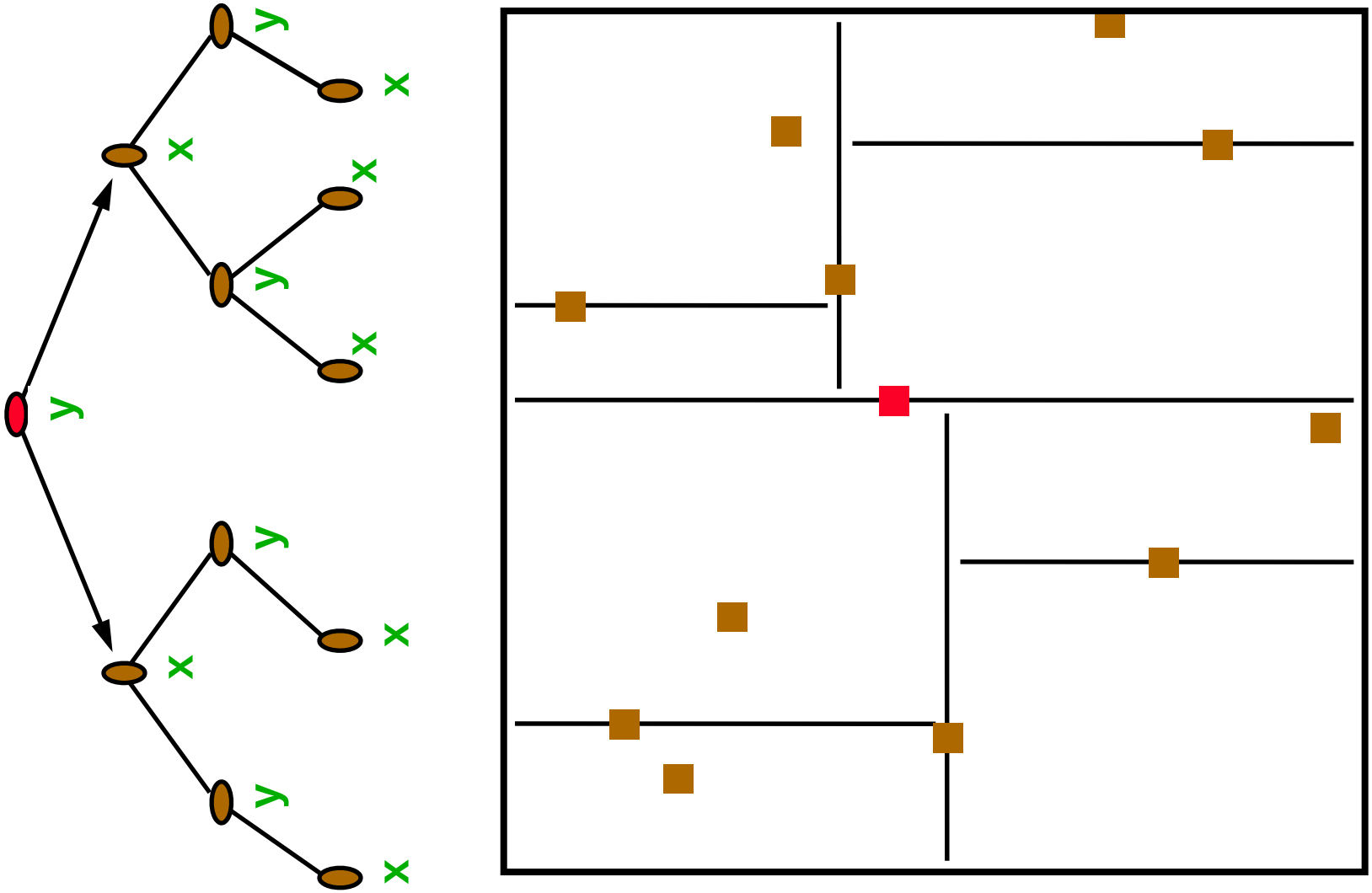
Offline



KD-stromy

- Budem využívat prostorovou hierarchii vybudovanou nad objekty scény.
- Algoritmus nevyžaduje konkrétní hierarchii, pro jednoduchost se zaměříme na KD-stromy.
- Uzel KD-stromu odpovídá osově orientovanému bounding boxu.
- Každý list stromu bude obsahovat seznam referencí na objekty protínající jeho odpovídající box.

KD-stromy



Hardwareový okluzní dotaz

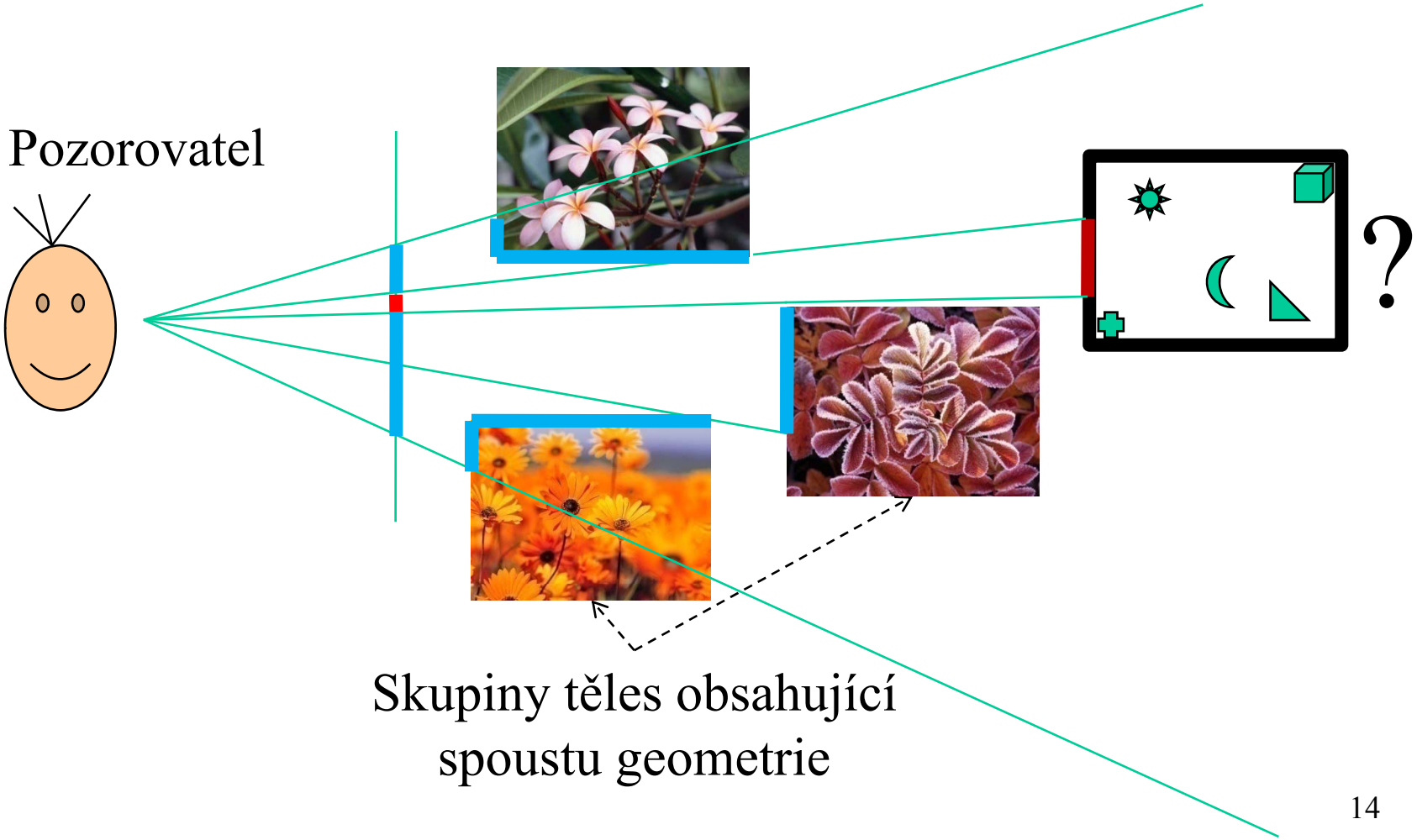
- Occludee – Potenciálně zakrytá část scény.
- Occluder – Těleso zakrývající část scény.

- Viditelnost části scény (Occludee) budeme testovat posláním jejího obalového tělesa do GPU. Ta nám pak vrátí počet viditelných fragmentů obalového tělesa vzhledem kestávající scéně. Tomuto dotazu říkáme okluzní dotaz (occlusion query).

Hardwareový okluzní dotaz

- Inicializuj dotaz;
- ...
- dotaz → start();
- vykresli objekt (bounding box ...);
- dotaz → end();
- ...
- Zjisti, kolik pixelů je viditelných;

Hardwareový okluzní dotaz

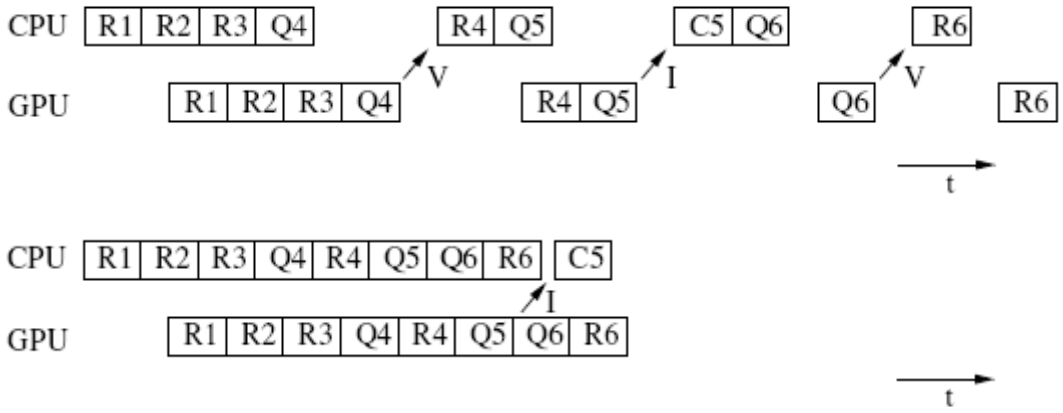


Hardwareový okluzní dotaz

- Výhody okluzního dotazu
  - Obecnost occluderů (můžeme použít libovolnou vykreslenou scénu).
  - Fůze occluderů.
  - Obecnost occludee (můžeme použít libovolná obalová tělesa – cokoli, co jde rychle převést do rasteru).
  - Plné využití moderní GPU.
  - Jednoduché použití.

Hardwareový okluzní dotaz

- Problém okluzního dotazu
  - Zpoždění mezi zadáním dotazu a obdržením výsledku → naivní implementace může být velmi pomalá.

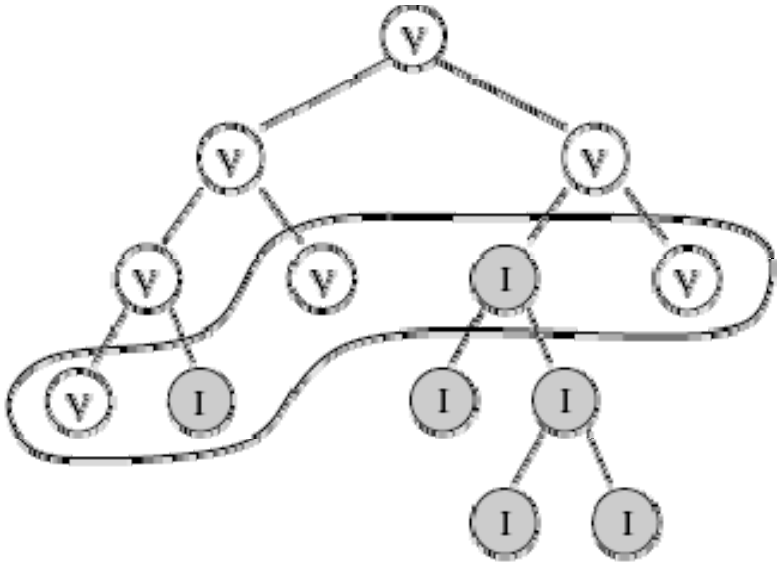


Původní algoritmus CHC (Coherent Hierarchical Culling)

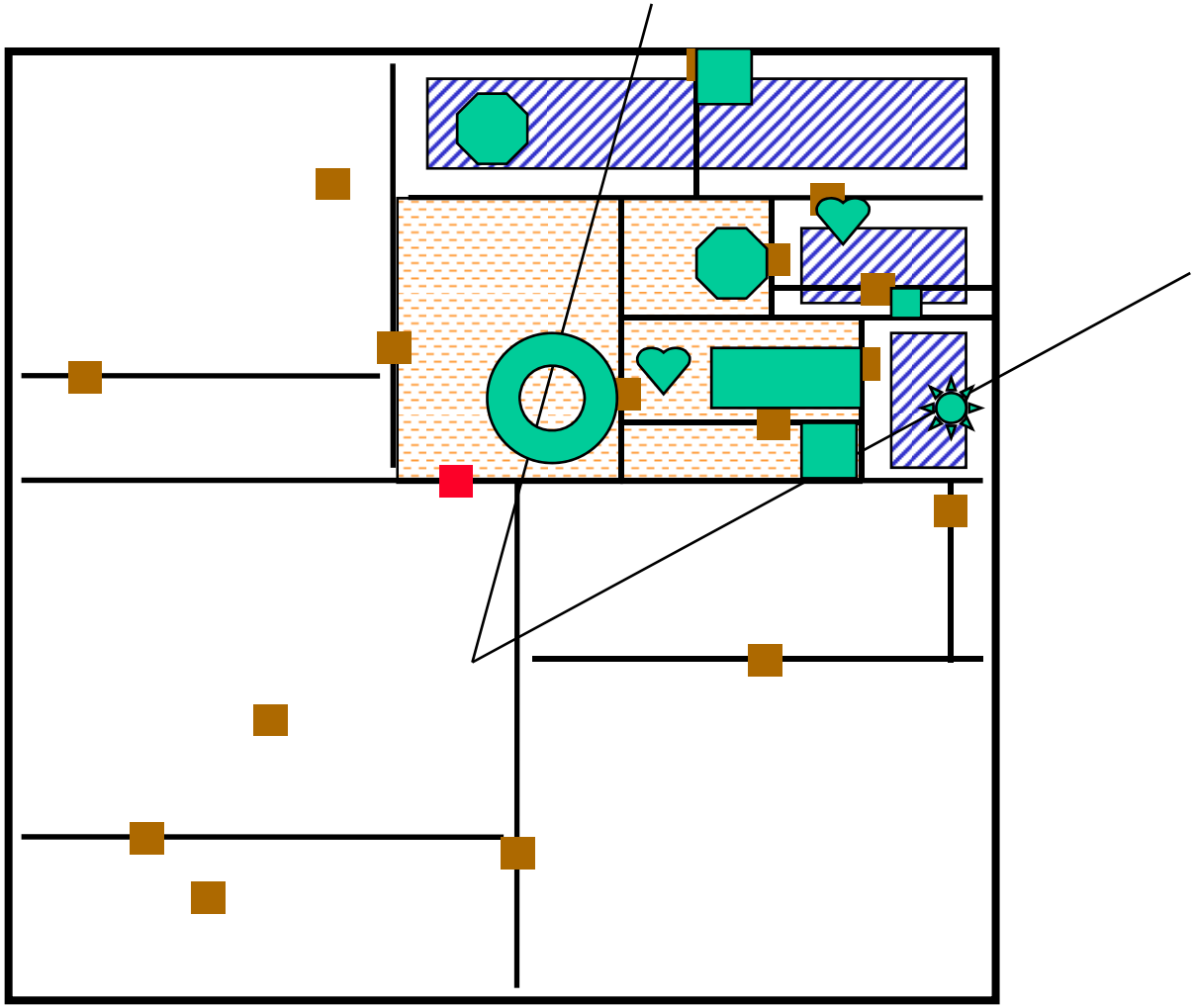
- Okluzní dotazy vyvoláme na uzlech hierarchie, kde průchod skončil v minulém kole. Tak se vyhneme dotazům na všech minule viditelných *vnitřních* uzlech.
- Předpokládáme, že dříve viditelné uzly zůstanou viditelné a vykreslíme jejich geometrii bez čekání na odpovídající okluzní dotaz.

CHC

- Uzly, které jsou listy, nebo jsou v hierarchii první neviditelné, nazýváme *koncové uzly*.
- Vnitřní uzly, které byly klasifikovány jako viditelné nazýváme *otevřené uzly*.



CHC



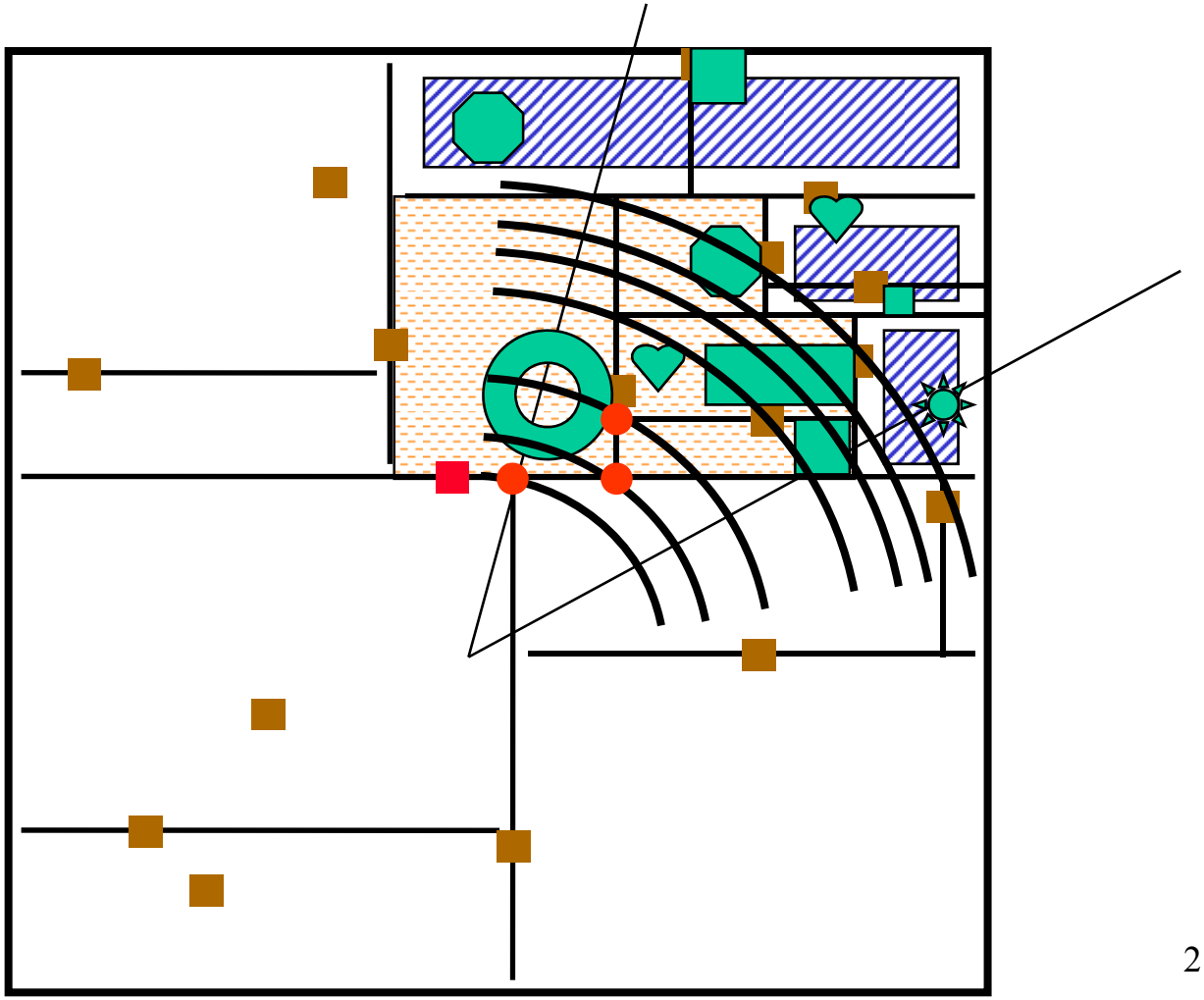
CHC

- Uzly scény zpracováváme podle vzrůstající vzdálenosti od pozorovatele
- V algoritmu nejprve uzly "začerníme" (označíme jakožto neviditelné) a pak při vyřešení viditelnosti propagujeme viditelnost směrem nahoru.

Algoritmus:

- Pokud jsou nějaké nezpracované okluzní dotazy, zpracuj je.
- Pokud nejsme hotovi, pokračuj požadavky na vykreslení viditelných uzlů a test původních *koncových* uzlů. Pak se vrať na předchozí bod.

CHC



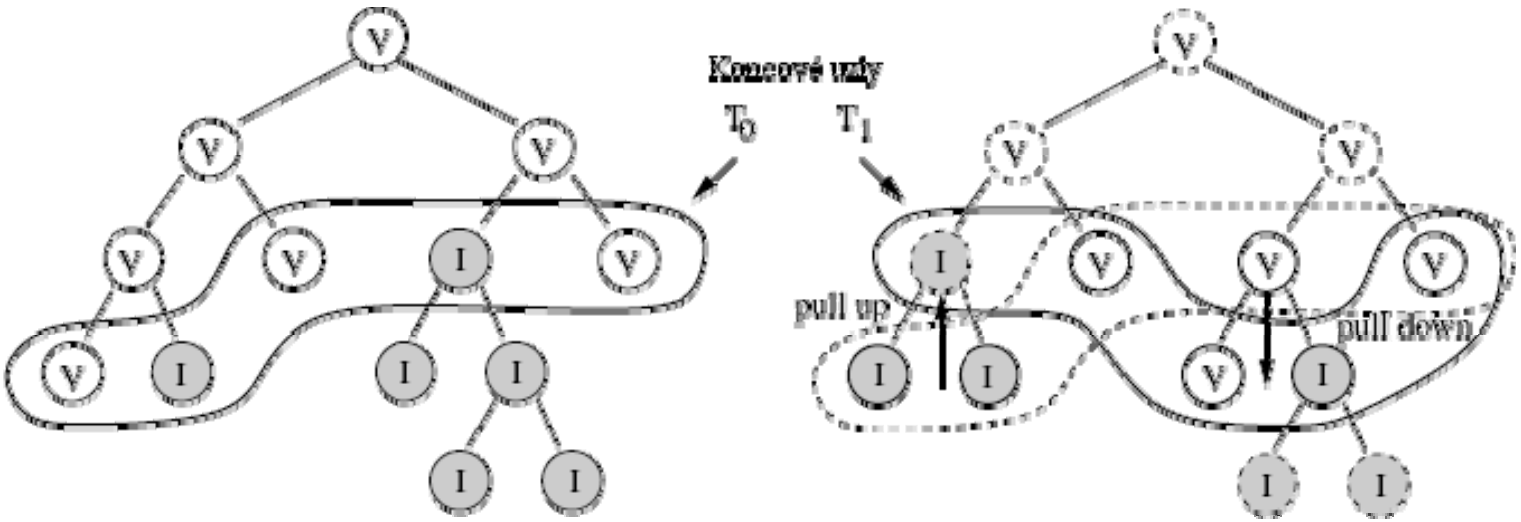
CHC

```
DistanceQueue.Push(root);
 while !DistanceQueue.Empty() || !QueryQueue.Empty() do
   while !QureryQueue.Empty() do
     if FirstQueryFinished() then
       N= QueryQueue.Dequeue(); HandleReturnedQuery(N);
   if !DistanceQueue.Empty() then
     N= QueryQueue.Dequeue();
     if InsideViewFrustum(N) then
       if !WasVisible(N) then
         QueryPreviouslyInvisibleNode(N);
       else
         TraverseNode(N);
```

```
TraverseNode(N)
begin
  if IsLeaf(N) then
    RenderLeaf(N);
  else
    DistanceQueue.PushChildren(N);
    N.IsVisible= false;
end
```

```
HandleReturnedQuery(N)
begin
  if Q.VisiblePixels > threshold then
    if !WasVisible(N) then
      TraverseNode(N);
    PullUpVisibility(N);
  else
      N.IsVisible= false;
end
```

CHC



CHC Optimalizace a problémy

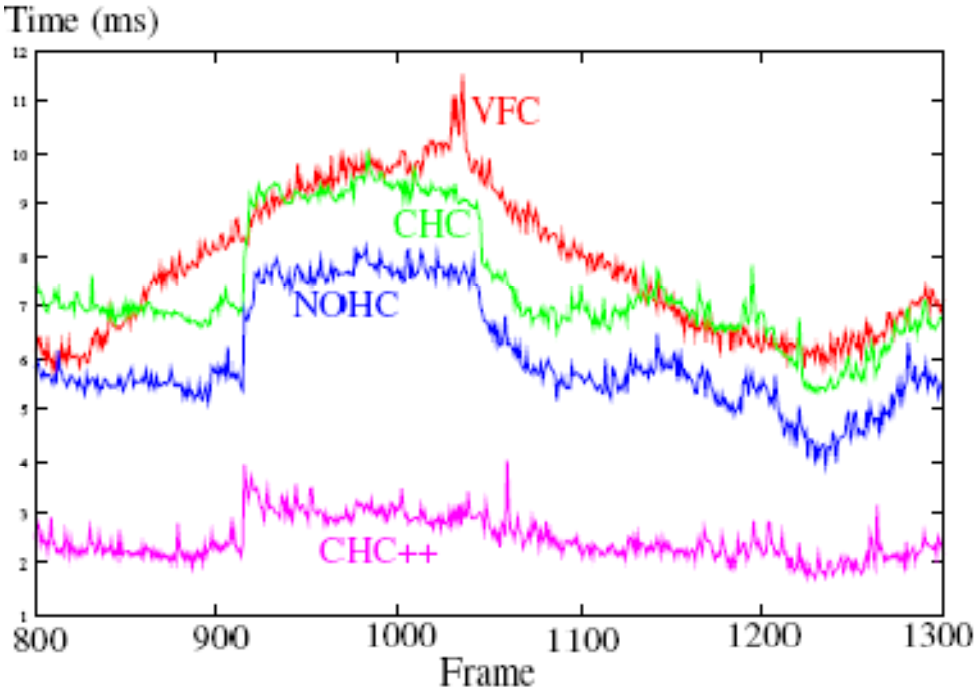
- Konzervativní testování viditelnosti
  - Předpokládáme, že kdykoli je uzel viditelný, zůstane viditelný po několik snímků.
- Aproximativní řešení viditelnosti (*threshold* > *n*)
- Kompletní odstranění prostojů CPU spekulativním vykreslováním některých uzlů s doposud nedořešenou viditelností.

- Základním problémem CHC, který vedl k CHC++ bylo časté přepínání stavů grafické pipeline. To vedlo k výrazně nižšímu výkonu.

CHC ++

- VFC - View Frustum Culling
- CHC - Coherent Hierarchical Culling
- NOHC - Near Optimal Hierarchical Culling
- CHC++ - Nová verze CHC

Model elektrárny,
12 000 000 trojúhelníků



CHC++

- Hlavní příspěvky CHC++ :
  - Redukce změn stavů (Fronty s dávkováním dotazů)
  - Redukce počtu dotazů (Multidotazy)
  - Redukce zdržení CPU (Zavedení náhody pro počet nedotázaných uzlů)
  - Redukce vykreslované geometrie (Těsné obalové objemy)
  - Možná integrace do herních "Engine" (Zavedení vykreslovací fronty).

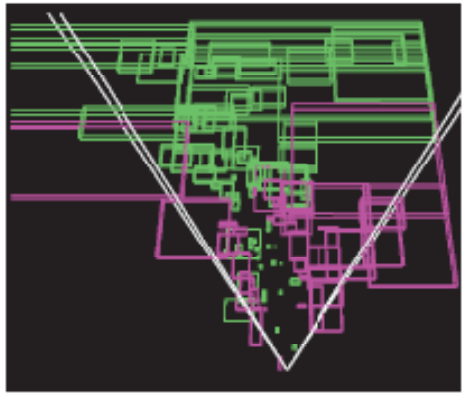
CHC++

Počet barev je počet změn stavů

Scéna

CHC

CHC++



CHC++

- Časté změny stavů v hardwareové pipeline jsou důvodem výrazně nižšího výkonu algoritmu
  - Zdržení je způsobeno na straně hardware (čištění bufferů), driverů i aplikace.
  - Vhodný počet je kolem 200 změn stavů na snímek.

- Tento problém se řeší dávkováním a zavedením "*viditelné*" a "*neviditelné*" fronty pro dříve videtelné (neviditelné) uzly.

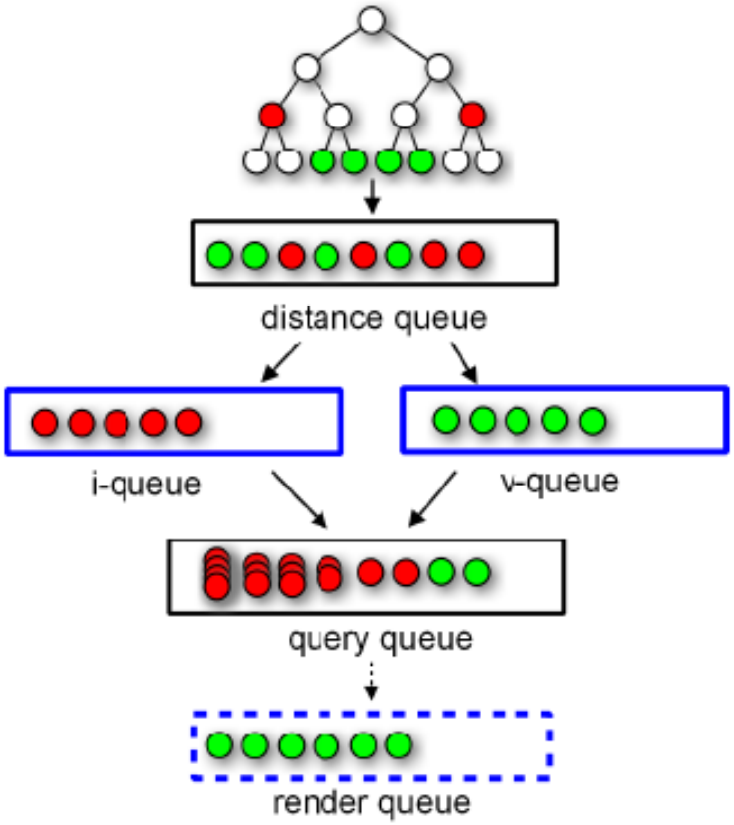
CHC++

- Jestliže počet uzlů v dávce pro *neviditelnou* frontu dosáhne předepsaného stavu (20 - 80), změníme stav na dotaz a zpracujeme je najednou.

- Také uzly ve *viditelné* frontě se zpracovávají dávkově. Avšak toto zpracování není kritické, protože geometrie se vykresluje jakožto první a výsledek dotazu bude důležitý až v následujícím snímku.

CHC++

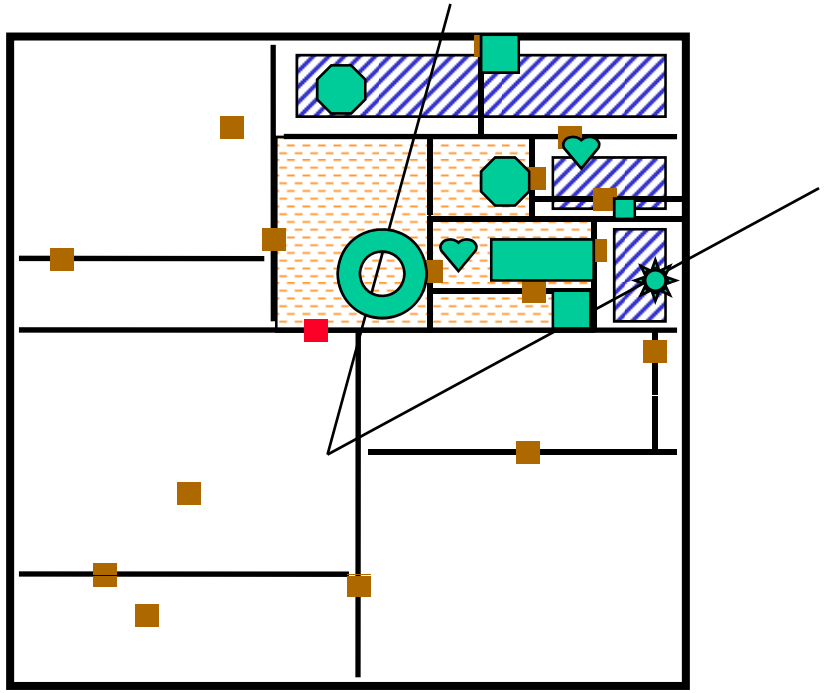
Pro snadné zapojení do herních engine je zde navíc ještě "vykreslovací" fronta shromažďující uzly, které budou vykresleny. Je zpracována v okamžiku, kdy má být vypuštěna dávka z nějaké fronty ke zpracování.



CHC++

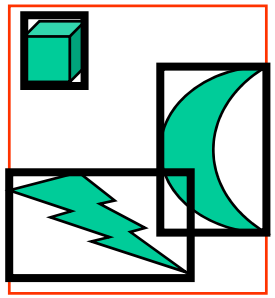
Multidotaz je dotaz na více uzlů najednou

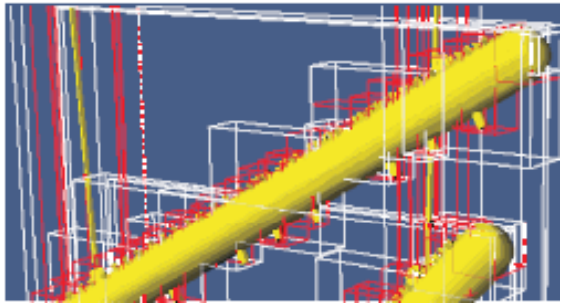
- Multidotazy je možné zavést pro uzly, které mají vysokou pravděpodobnost, že nebudou vykresleny.
- Místo jednoho dotazu zadáme pouze jeden



CHC++

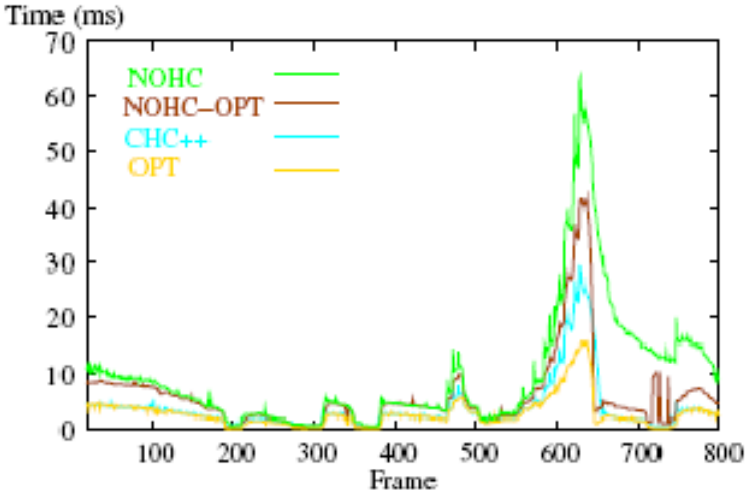
- Myšlenka "Těsných obalů" je založena na tom, že obal zde může být složen z několika jednoduchých těles
- Jako obal uzlu tedy vezmeme obalová tělesa jeho potomků.





Výsledky

Porovnání s algoritmem kreslícím pouze viditelné uzly



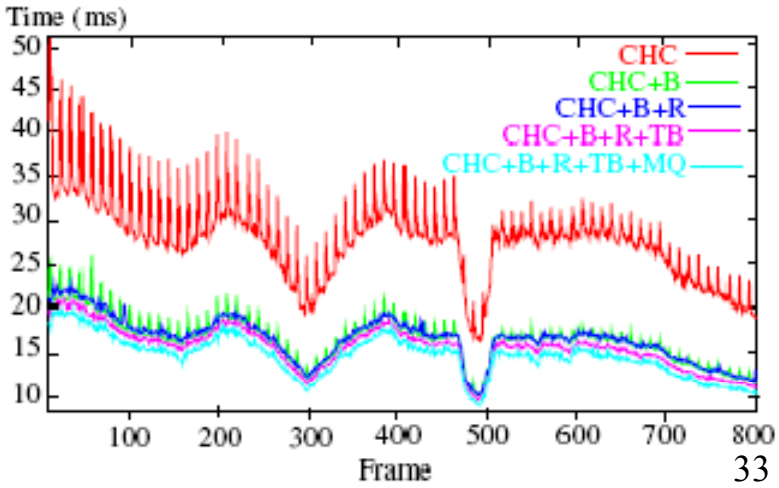
Postupné přidávání optimalizací

B - Batching

R - Randomization

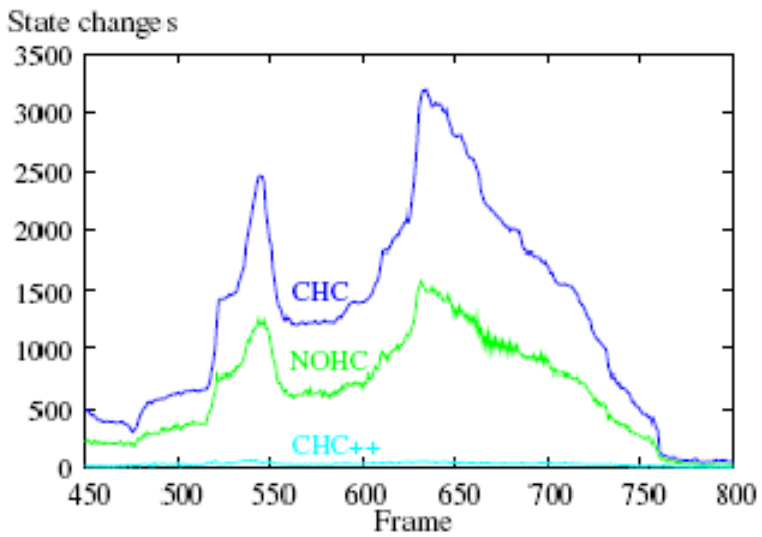
TB - tighter bounds

MQ - Multiqueries

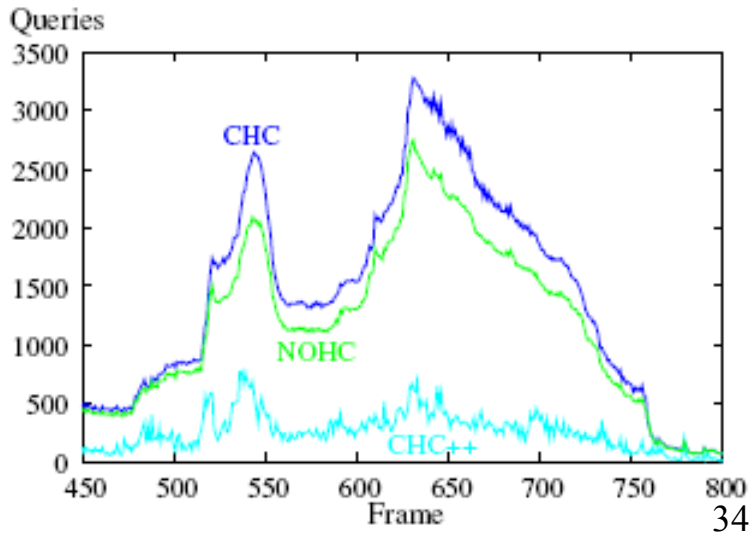


Výsledky

Snížení změn počtu stavů



Snížení počtu stavů



Použitá literatura

- *Visibility in computer graphics*, Bittner J., Wonka P., Environment and Planning, 2003
- Coherent hierarchical culling: Hardware Occlusion Queries made useful, Bittner J., Wimmer M., Piringer H., Purgathofer W., Eurographics 2004
- *Near optimal hierarchical culling: Performance driven use of hardware occlusion queries*, Guthe M., Balazs A., Klein R., Eurographics Symposium on Rendering 2006
- *CHC++ : Coherent Hierarchical Culling Revisited*, Oliver Mattausch, Jiří Bittner, Michael Wimmer, Eurographics 2008