

Hierarchie v PG

- často komplexní scény, komplexní objekty, vzťahy
- snažíme se zachytit / zobrazit pohled do modelovaného světa, v omezeném čase (10-20 ms / 100-50 fps)

S omezenými prostředky (GPU RAM)

→ nemůžeme věnovat čas každému modelu / Δ / vertexu, atd. ve scéně, při plné komplexnosti detailů

Poučení z historie



Což takhle ji, dobýt.... ?

Jak zajistíme kontrolu nad územím ?

Poučení z historie



- Belgica Prima
- Belgica Secunda

- Lugdunensis Prima
- Lugdunensis Secunda
- Lugdunensis Tertia
- Lugdunensis Quarta

Rozdělení a panství

- Velké území → menší celky
- Loajální samospráva
- Podpora izolace

Poučení z historie

Císař

Senát

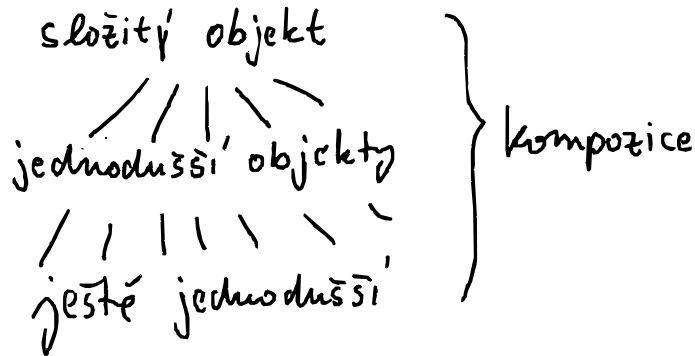
Guvernéři provincií

Mistři elity

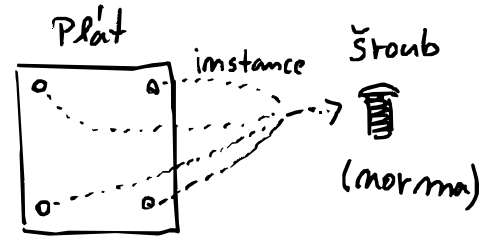
Vojáci, legie

Prostí občane'

Hierarchické 3D modelování



Př.



Opakování prvků – instance
(sdílené části) + variabilita

- atributy
- transformace
- materialy

Database 3D objektů

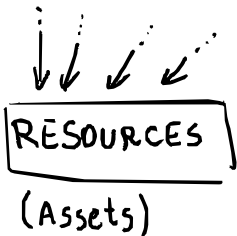
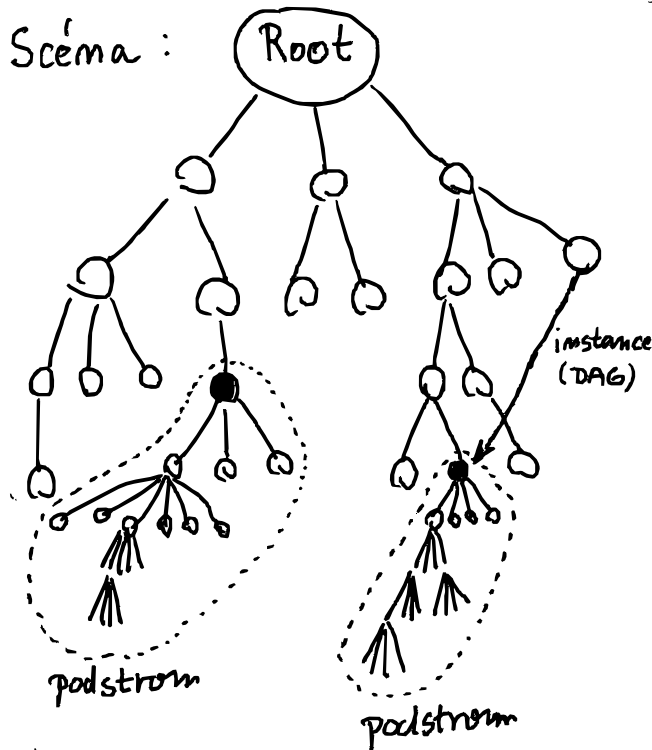
- strojírenství
- architektura
- game-dev

Hierarchické 3D formáty

- Open Inventor
 - Performer
 - VRML
 - OpenSG, X3D
 - PoV Ray
 - glTF
 - ...
- } hierarchická madstavba nad OpenGL
- pro virtuální realitu, web

Scene graph

Scéma :

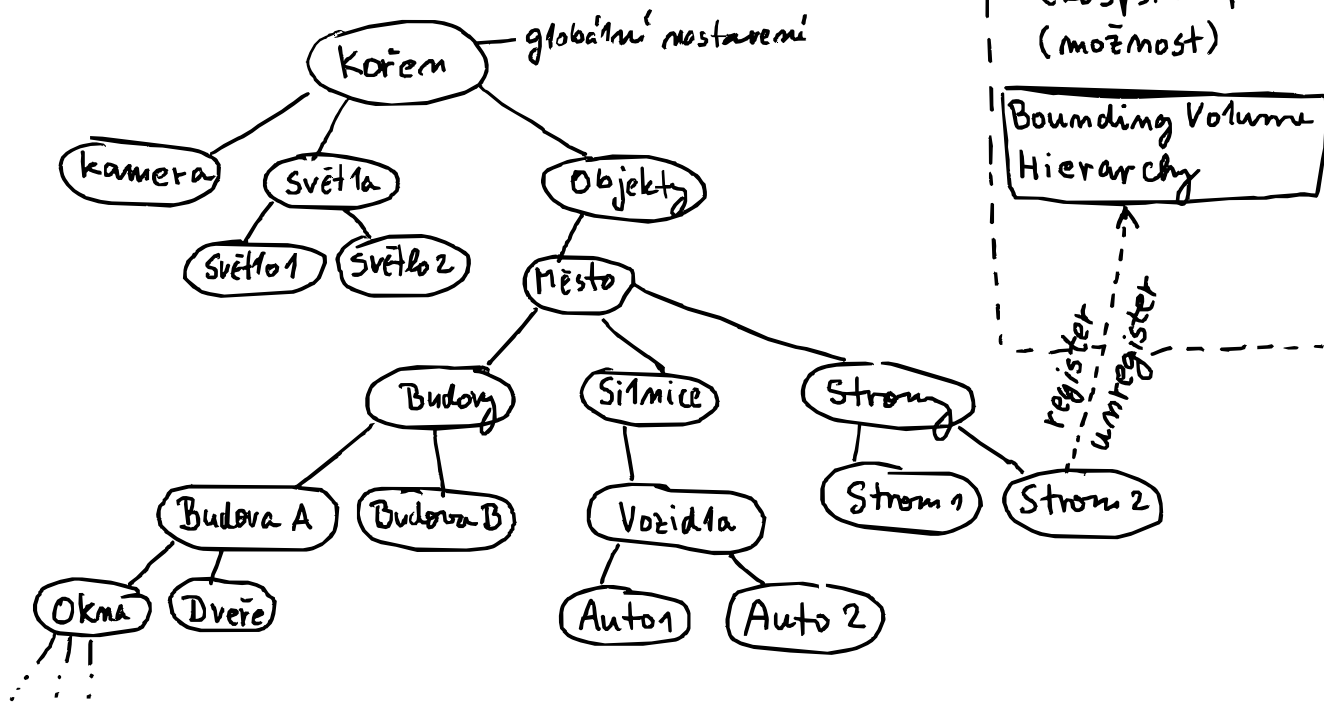


DAG \rightarrow strom

Hierarchická struktura (DAG)

- zachycuje logickou strukturu pro transformace a rendering
- nadřazený / podřazený uzel
- propagace efektu uzlu z rodiče na potomky (podstrom) např. transformace (pozice a matice objektu ve světě, měřítko, ...)
- různé typy uzlů
 - geometrie zpravidla v listech
- rozšiřitelnost, konfigurace
- komunikace / signály
- Výsledek dán průchodem grafu od kořene k listům

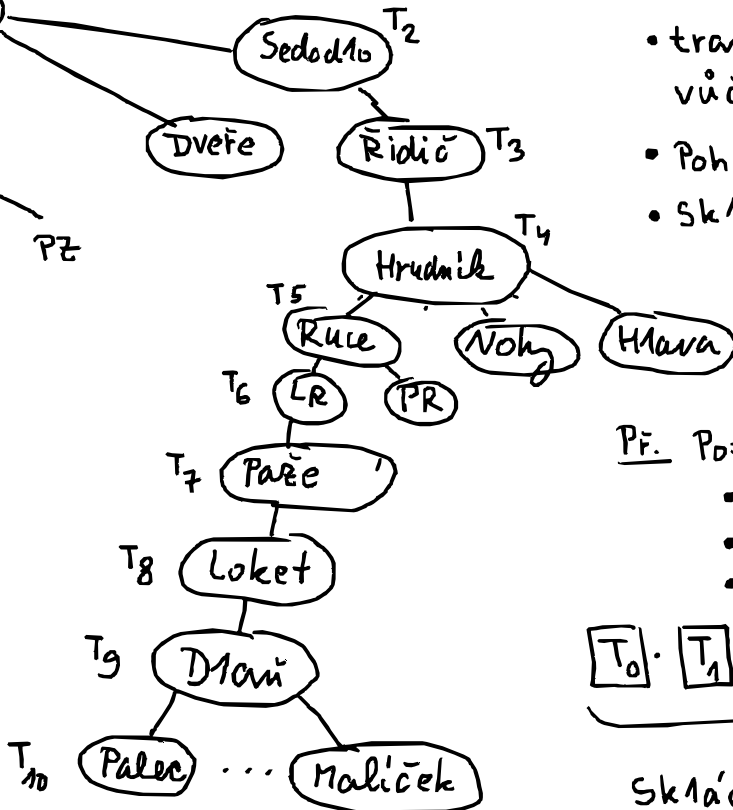
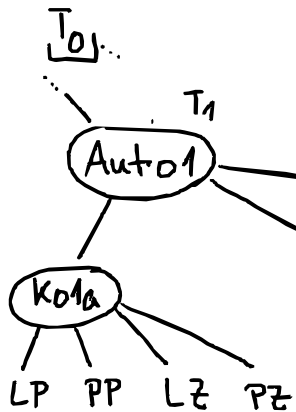
Příklad - logické celky



Průchod stromem při různých příležitostech

- mapř. událost render()
- navštěvování uzlů do hloubky, pre-order traversal
- me nutně všech (uzly mimo záběr, deaktivované podstromy, exkluzivně vyselektované pokračování do potomka, ...)

(Hypotetický) Příklad



Pozice objektu (uzlu)

- transformace relativní vůči rodičovskému uzlu
- Pohmu autem → pohmu řidičem
- skládání transformací

Př. Pozice palce ve světě:

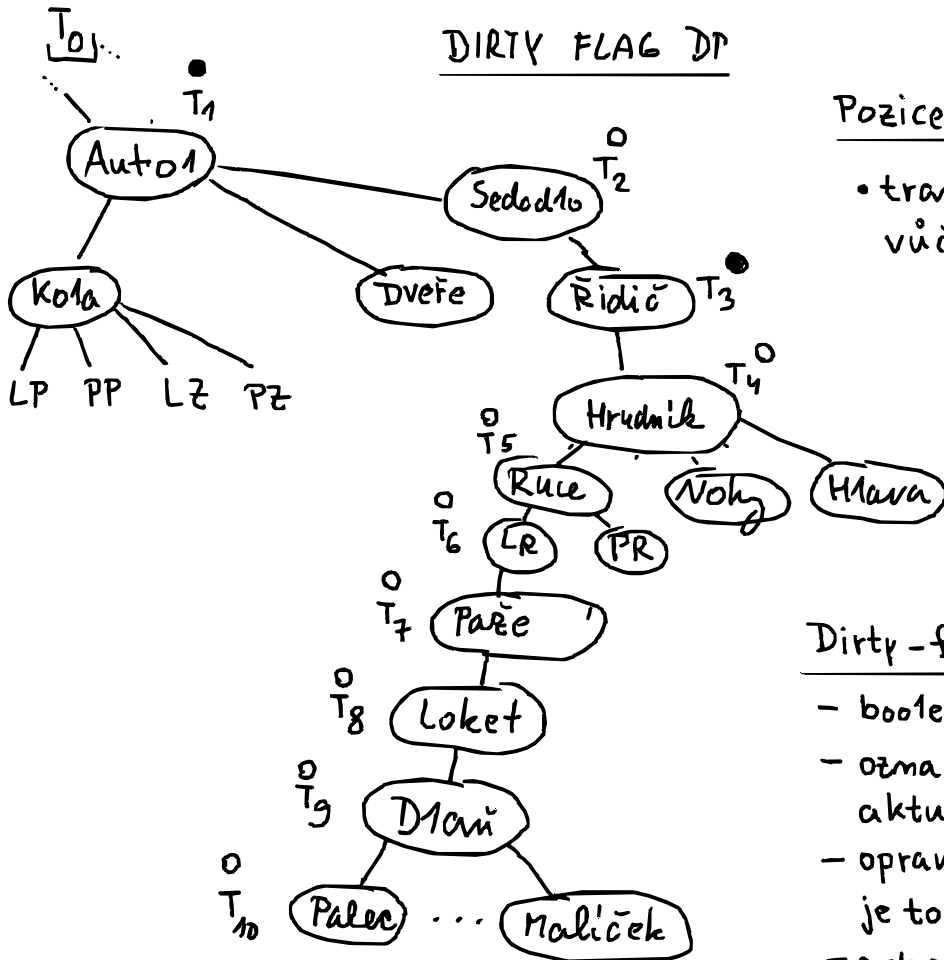
- třeba pro rendering
- nebo pro test viditelnosti
- nebo pro test kolizí (?)

$$\boxed{T_0} \cdot \boxed{T_1} \cdots \boxed{T_8} \cdot \boxed{T_9} \cdot \boxed{T_{10}} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Skládání transformací

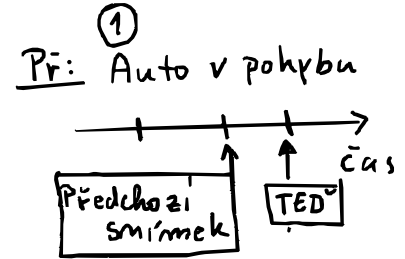
- cesta od uzlu ke kořeni
- násobení matic 4x4

DIRTY FLAG DP



Pozice objektu (uzlu)

- transformace relativní vůči rodičovskému uzlu



Dirty-flag design pattern

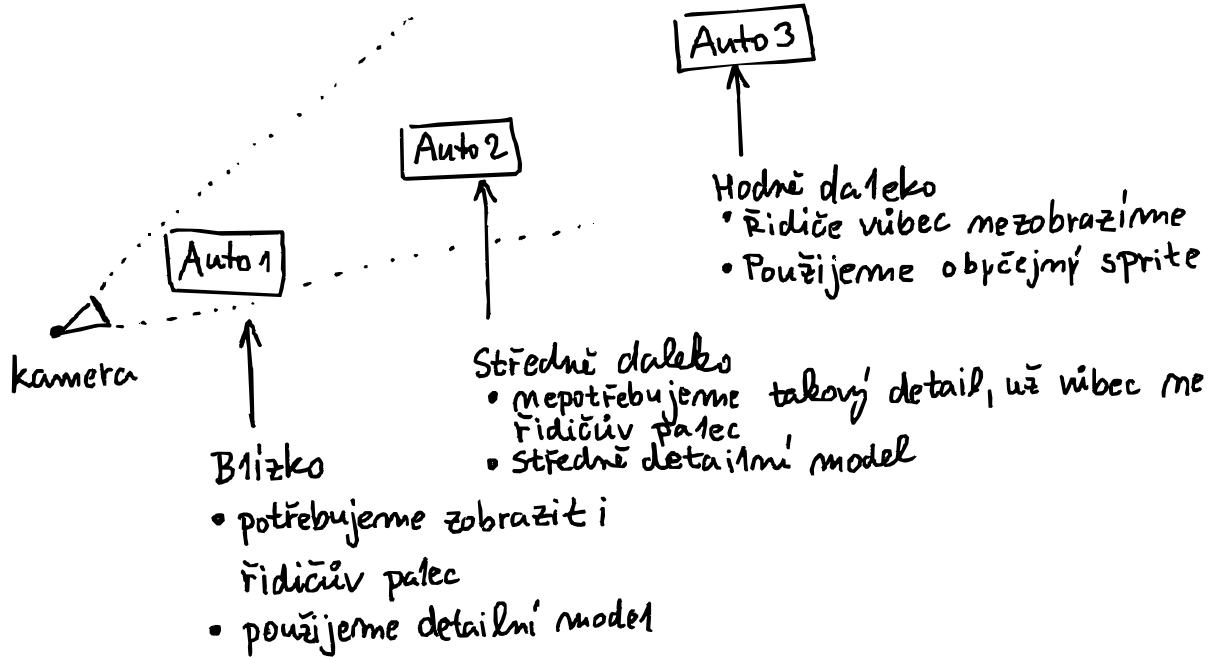
- boolean flag dirty
- označení, že data nejsou aktuální (dirty = true)
- oprava/přepočítání, až když je to potřeba (on-demand)
- nebo při "vhodné příležitosti"
- a nastavení dirty = false
- např. při dalším průchodu stromem
- dirty auto → dirty palec

Průchod stromem

→ výpočet pozice auta, sedadla, řidiče, bounding box mimo frustum, metřeba pokračovat až k palci.

Redukce komplexnosti scény

- podstromy mohou být různě komplexní
- pro většímu objektů nepotřebujeme plný detail
- jen pro ty, které jsou dostatečně blízko aktivní kamery



Redukce komplexnosti scény

- podstromy mohou být různě komplexní
- pro většímu objektu nepotřebujeme plný detail
- jen pro ty, které jsou dostatečně blízko aktivní kamery

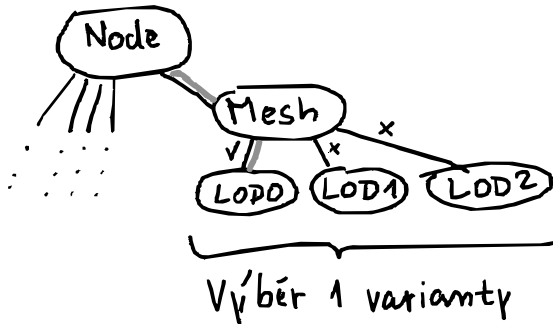
Level of Detail / Úroveň detailu

- nahradíme složitou trojúhelníkovou sítí jednodušší
- bude se renderovat méně → rychleji
- např.:

LOD0: nejvyšší detaily

LOD1: střední

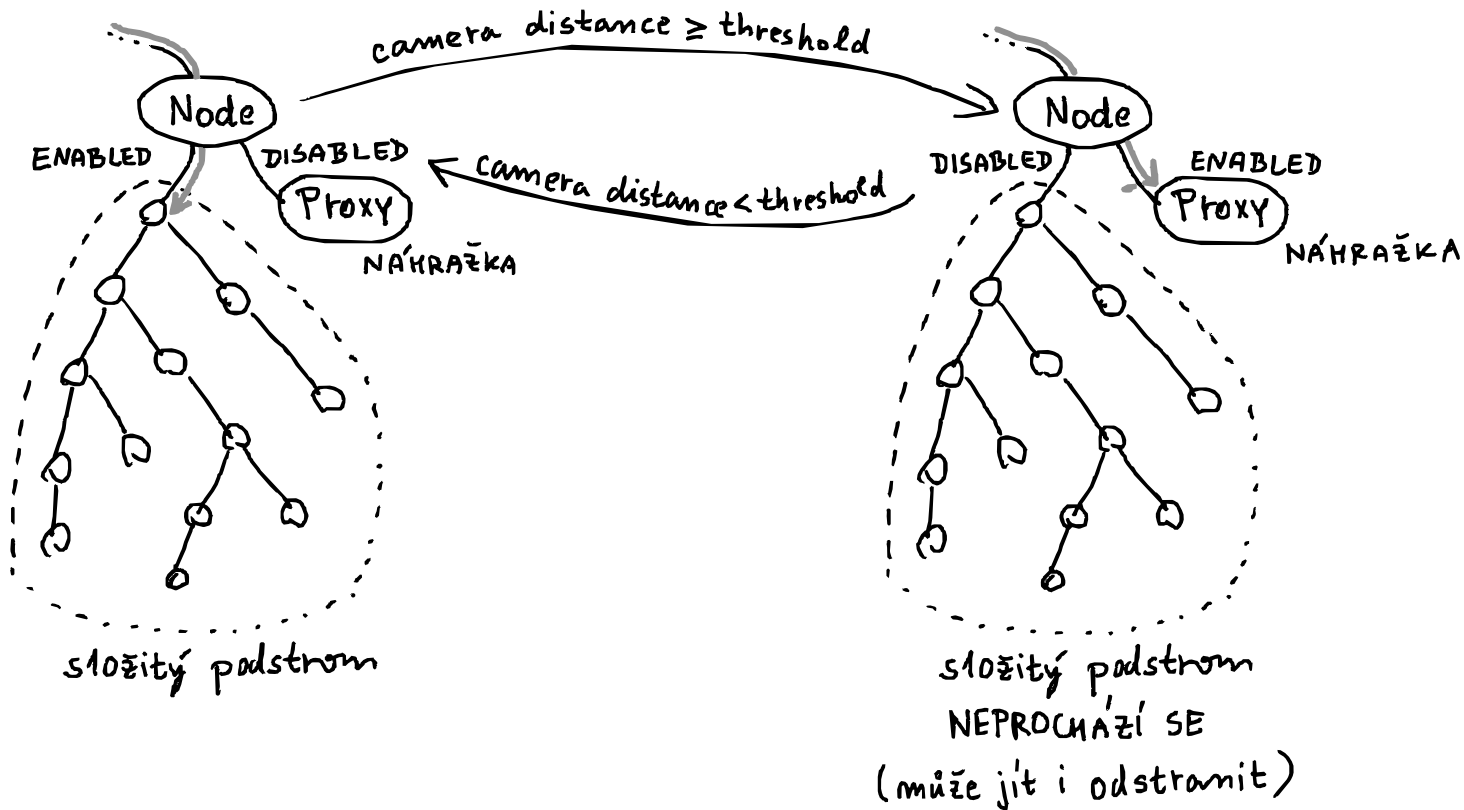
LOD2: nejnižší



Adaptivně přepínáme mesh, který se bude renderovat v další smítku. Data pravděpodobně už má GPU, jen zvolíme jiné ID bufferu.

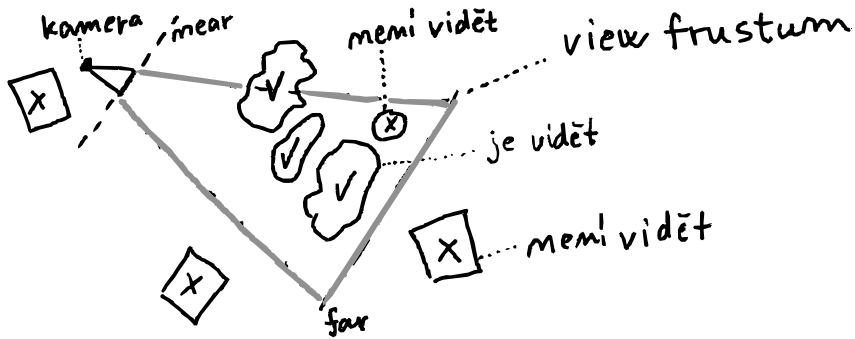
Redukce komplexnosti scény - PROXY

- Nahradíme celý podstrom zástupným uzlem
- Podstrom se nemusí procházet
- Přepínáme automaticky (podobně jako LOD)



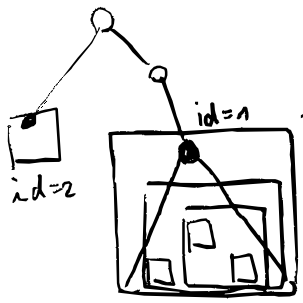
Viditelnost

- Mmoho objektů nemusí být vůbec vidět (nemusíme je renderovat)
 - mimo pohledový objem
 - zastíněné jinými objekty (occlusion)



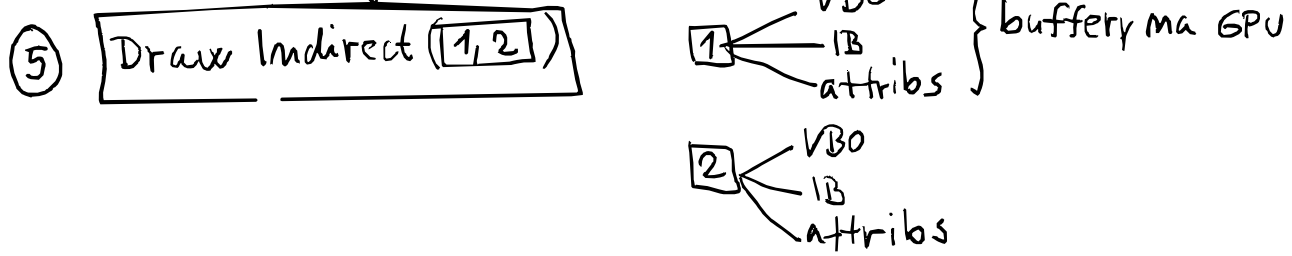
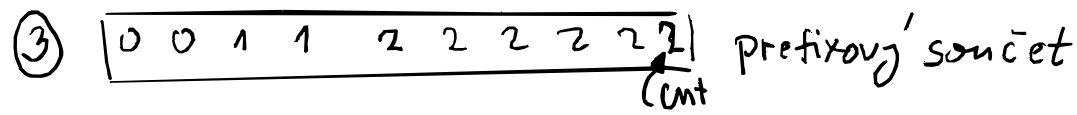
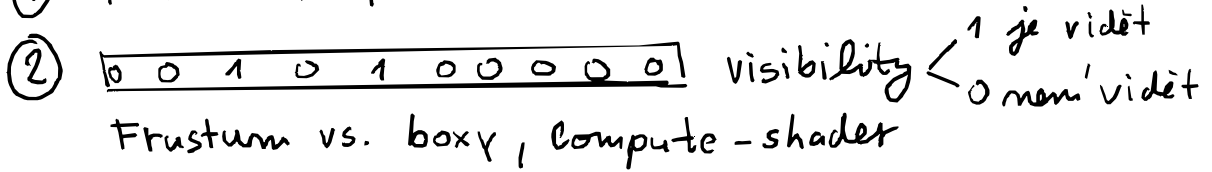
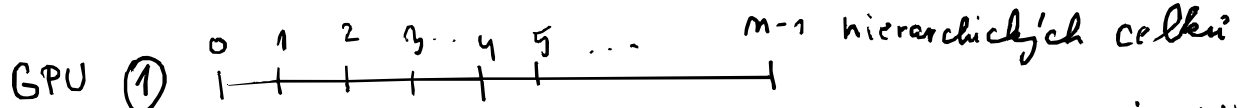
- Viditelnost za nás do jisté míry řeší GPU
 - z-buffer
 - frustum culling - ořezání geometrie (+occlusion queries)
 - zpravidla to nestačí.
- Na úrovni grafu scény si musíme pomoci sami
 - resp. pomůže použitý engine

Visibility - testy na GPU



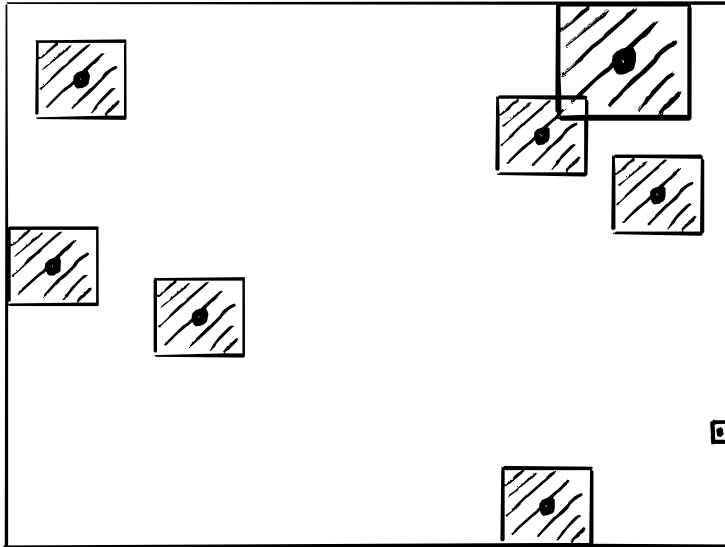
Bounding box hierarchického celku (podstromu)

→ id objektu, bounding box

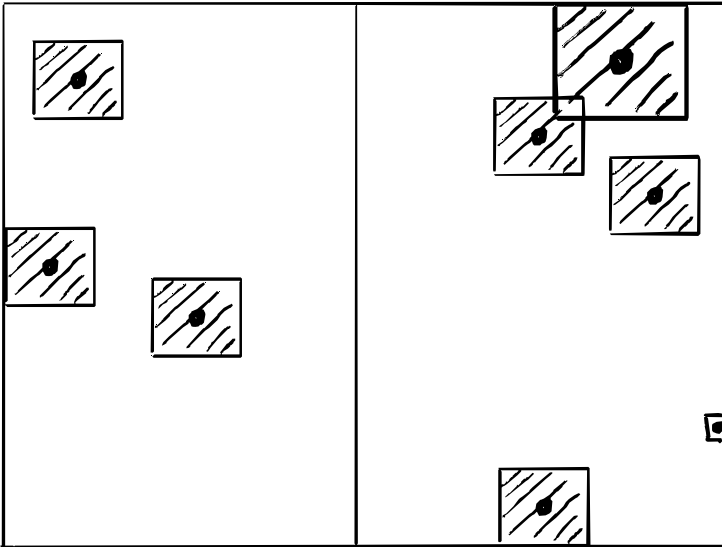


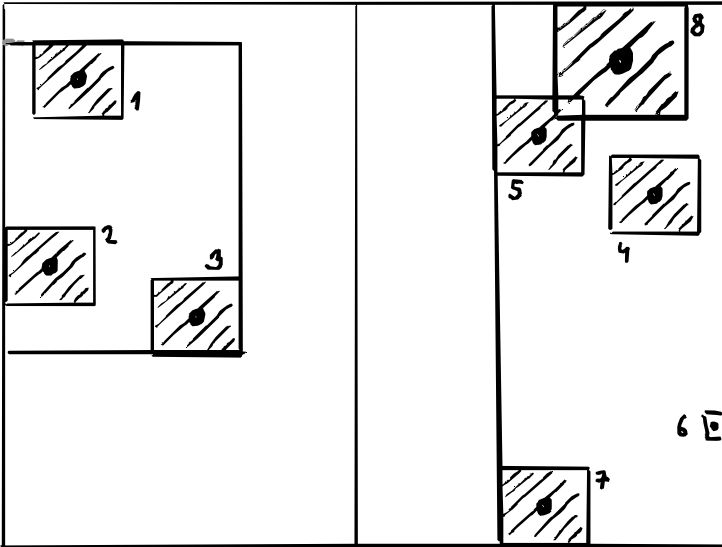
(BVH základní struktura) - Draft

① AABB



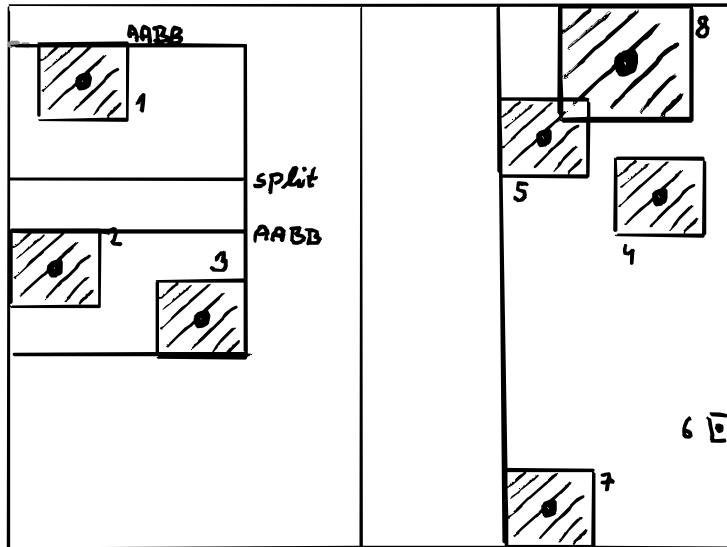
② SPLIT



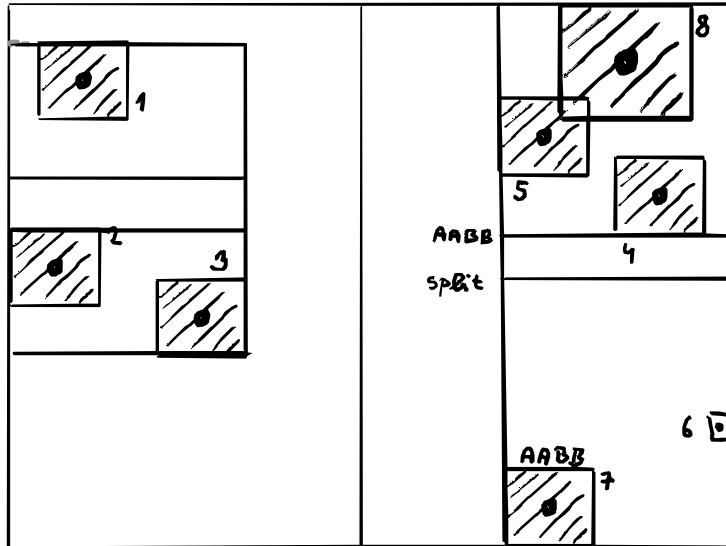


③ PARTITION $\left\{ \begin{array}{l} \text{LEFT} \\ \text{RIGHT} \end{array} \right.$

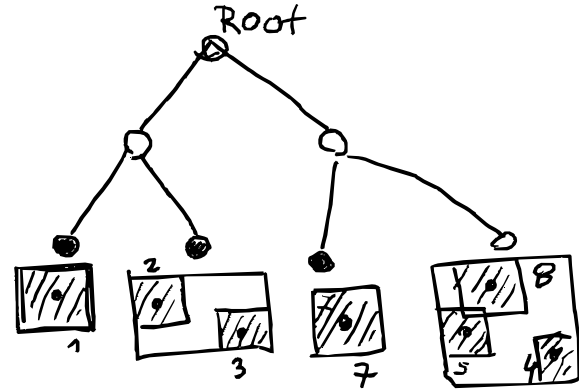
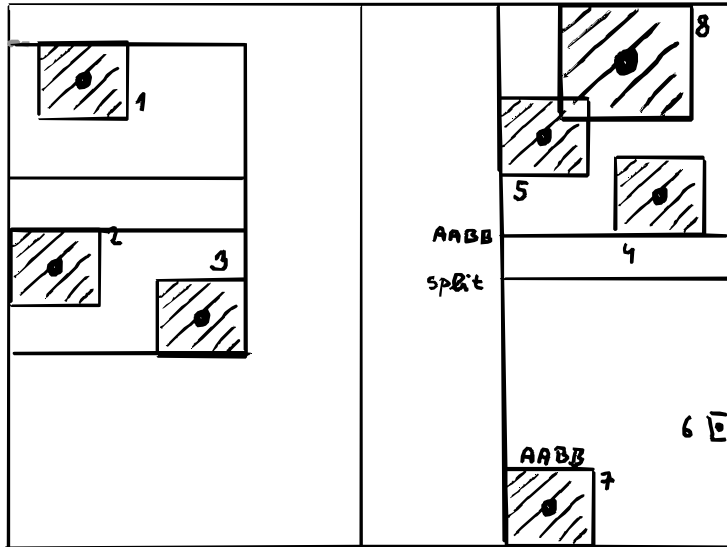
④ A A B B (LEFT)
A A B B (RIGHT)



- ③ PARTITION $\left\{ \begin{array}{l} \text{LEFT} \\ \text{RIGHT} \end{array} \right.$
- ④ AABB (LEFT)
AABB (RIGHT)
- ⑤ SPLIT LEFT
- ⑥ PARTITION
- ⑦ AABB



- ③ PARTITION $\left\{ \begin{array}{l} \text{LEFT} \\ \text{RIGHT} \end{array} \right.$
- ④ AABB (LEFT)
AABB (RIGHT)
- ⑤ SPLIT LEFT
- ⑥ PARTITION
- ⑦ AABB
- ⑧ SPLIT RIGHT
- ⑨ PARTITION
- ⑩ AABB



Split - strategie

- pomocí poloviny nejširšího rozměru AABB
- surface area heuristic (SAH)
- minimalizace překryvu medián

partition - přečišlování objektů vlevo/vpravo
 - rozhodnutí vlevo/vpravo např. podle středu AABB