Graf scény a dátové štruktúry

Zara kap. 14

1 Úvod

- Nezobrazovane objekty
- Zobrazovane objekty
- Prvky definujúce log. štruktúru scény
- Transformácie

Prehľad:
- Graf scény
- Pomocne dátové štruktúry
    - Hierarchie obálok - Bounding Volume Hierarchies (BVH)
    - Hierarchie delenia priestoru
        - Binary Space Partitioning Trees (BSP strom)
        - Octrees (Oktalovy strom)
        - kD strom

2 Graf sceny

3 Pomocne datove struktury

- Počiatočná fáza – vytváranie a napĺňanie
- Ďalšia fáza – používanie na vyhľadávanie

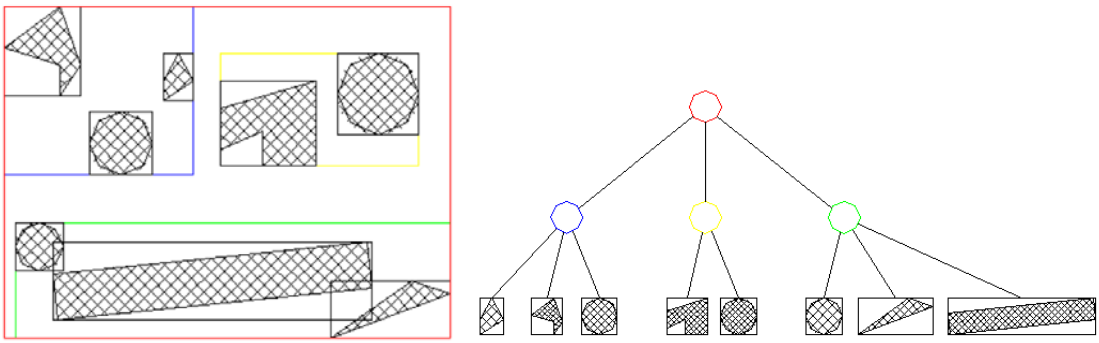
Kategórie:
- Hierarchie obálok –
  Bounding Volume Hierarchies (BVH)
- Hierarchie delenia priestoru –
  Space Subdivision Hierarchies

3.1 *Hierarchie obálok*

V scénach s veľkým počtom objektov, kde sú úplne viditeľné a úplne neviditeľné objekty zoskupené do niekoľkých priestorových skupín (týchto skupín môžu byť aj tisíce, avšak musí to byť rádovo menej ako je celkový počet objektov v scéne), nie je ani prístup ohraničujúcich objektov dostatočným zrýchlením. Príkladom takejto skupiny neviditeľných objektov je napríklad časť mesta, ktorá je celá skrytá kopcom alebo inou prekážkou. Príkladom viditeľnej skupiny je napríklad kruhové námestie, na ktorého začiatku stojí pozorovateľ, ktorý pozerá na toto námestie. V takýchto scénach sa úspešne používajú hierarchické testy viditeľnosti.

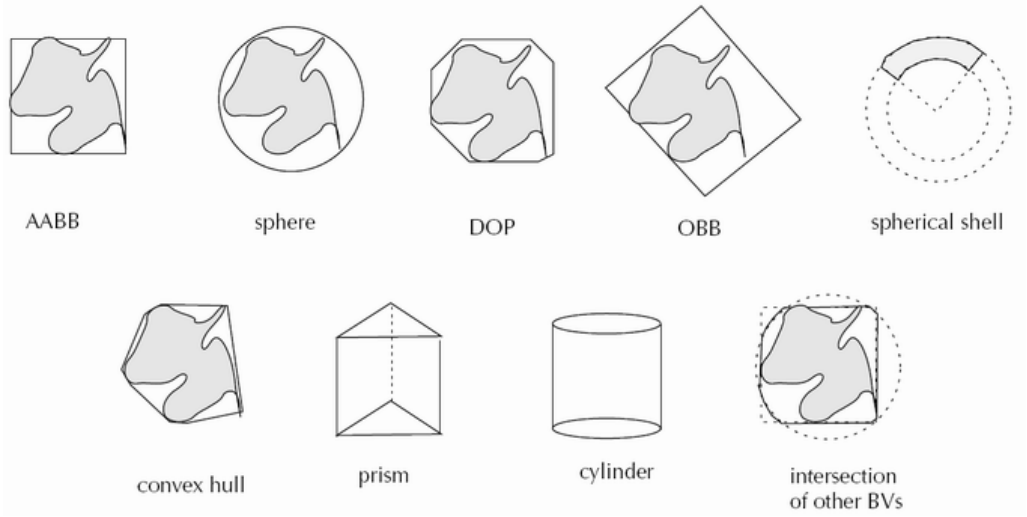
Najvrchnejší uzol stromu je koreň, ktorý nemá otca. Listy nemajú žiadnych potomkov. V každom liste je uložený objekt scény. Vnútorné uzly majú smerníky na ich synov. Každý

uzol, vrátane listov má svoje ohraničujúce teleso – preto sa táto hierarchia nazýva hierarchia ohraničujúcich telies. Ohraničujúce teleso uzla je ohraničujúcim telesom všetkých objektov, ktoré sú uložené v listoch podstromu s týmto uzlom ako koreňom. Často je požadované, aby toto teleso bolo ohraničujúcim telesom aj pre ohraničujúce telesá synov tohoto uzla. Teda ohraničujúce teleso uložené v koreni stromu je ohraničujúcim telesom všetkých objektov z tohoto stromu. Príklad BVH si môžeme pozrieť na nasledujúcom obrázku.



Ohraničenie objektu obálkou (bounding volume).

- Guľa -
- Osovo zarovnaný kváder (AABB)
- R-stromy
- Orientovaný kváder (OBB)
- Orientovaný rovnobeznosten (k-DOP)
- Konvexný obal



Samozrejme je možné, aby bol v každom uzle BVH použitý iný druh ohraničujúceho telesa. Ak sú však použité algoritmy optimalizované pre niektorý typ ohraničujúcich telies, tak sa najčastejšie používa iba tento jeden typ v celej hierarchii.

Ďalším problémom je vybudovať takúto hierarchiu. Problémy sú v podstate dvoch druhov:

1) vytvoriť vhodné ohraničujúce teleso pre dané objekty alebo ohraničujúce telesá
2) vhodne rozdeliť objekty do skupín, ktoré vytvoria ďalšie uzly hierarchie

Riešenie problému 1) je pre niektoré typy ohraničujúcich telies jednoduché a dá sa nájsť takmer v každej v literatúre, napríklad v Real-time rendering [3]. Pre niektoré typy BV je

možné vytvoriť viacero takýchto telies, pričom sa nedá vo všeobecnosti určiť, ktoré je vhodnejšie. Zložitejšie teleso je často menšie a presnejšie i keď za cenu časovo náročnejšej práce s ním. Príkladom takýchto ohraničujúcich telies sú k-DOPy. Problémom u takýchto typov BV je vybrať vhodný kompromis medzi presnosťou a zložitosťou.

Vo všeobecnosti ideálne riešenie problému 2) neexistuje. Blízko ku ideálnemu rozdeleniu objektov v uzle medzi synov je rozdelenie, ktoré by uzlu vytvorilo troch synov. V prvom by boli iba neviditeľné objekty, v druhom iba úplne viditeľné a v treťom čiastočne viditeľné objekty. Problém takéhoto rozdelenia je, že uzlom sa väčšinou nedá spraviť také ohraničujúce teleso, aby bolo efektívne zistiteľné, že je celé neviditeľné resp. viditeľné pre prvé dva prípady. Ďalším problémom je, že tieto tri množiny objektov sa menia s meniacou sa pozíciou a smerom pohľadu pozorovateľa a taktiež so zmenou pozície alebo tvaru objektov v scéne.

3.1.1 Guľa

Sphere tree.
Testovanie pozície gule vzhľadom na rovinu je jednoduché. Avšak vytvorenie ohraničujúcej gule pre dané teleso je netriviálny proces. Existuje množstvo algoritmov s rôznymi pomermi kvality ku rýchlosti. Jeden z najrýchlejších vypočíta AABB a jemu opísanú guľu. Táto guľa je hľadaná hraničná guľa. Je však zrejmé, že toto je horší výsledok ako samotné AABB.

3.1.2 Osovo zarovnaný kváder

Axis-Aligned Bounding Box (AABB)
Steny kvádra sú orientované tak, aby jeho objem bol čo najmenší.

Vlastnosť zarovnania ku osiam umožňuje mnohé testy robiť veľmi jednoducho. Napríklad pri teste pozície AABB a roviny stačí porovnávať jeden až dva vrcholy s rovinou (v závislosti od skutočnej pozície). Naviac informácia, o ktoré dva vrcholy AABB ide, je závislá iba od roviny samotnej, nie od AABB. Teda testovanie pozície veľkého počtu AABB s tou istou rovinou je veľmi rýchle.

3.1.3 Orientovaný kváder

Oriented Bounding Box (OBB)
OBB má tieto vlastnosti tiež, avšak iba v prípade, že majú všetky kvádre takú istú orientáciu. Zvoliť vhodnú orientáciu pre dynamickú scénu sa nedá. Teda vlastnosť orientácie nám nijako na efektivite vo všeobecnosti v dynamickej scéne nepridá. Naviac vytvorenie OBB pre daný objekt je zložitejší proces ako vytvoriť AABB. Aj preto je v dynamických scénach AABB vo všeobecnosti vhodnejší ako OBB.

3.1.4 Orientovaný rovnobeznosten (k-DOP)

k – Discrete Orientation Polytop (k-DOP)
Protiľahlé obálky sú rovnobežné a ich normály sú definované v *k* diskrétnych smeroch.

Najčastejšie varianty:
6-dop – kváder

14-dop – kváder s odseknutými rohmi
18-dop - kváder s odseknutými hranami

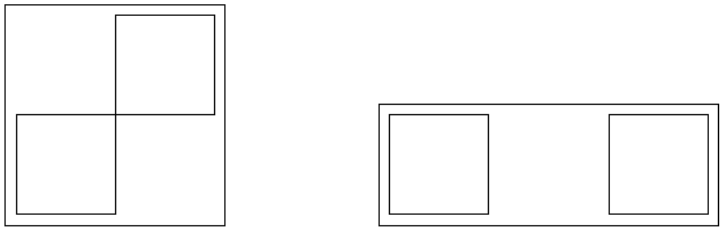
k-DOP je v porovnaní s predchádzajúcimi možnosťami veľmi zložitý objekt na testovanie pozície s inými jednoduchými telesami a rovinami. Jeho vytvorenie je tiež pmodstatne zložitejší proces. Preto nie je vhodným kandidátom pre dynamické scény, v ktorých sa objekty pohybujú, menia, pribúdajú nové a niektoré zanikajú.

3.1.5 Konštrukcia BVH

3 prístupy:
- Zdola – hore
- Zhora – dole
- Vloženie

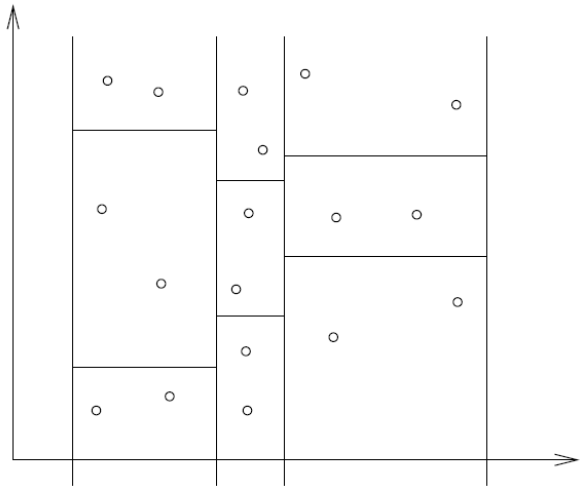
Zdola – hore.
Dva algoritmy:
1. Nech B je množina BV na najnižšej úrovni BV hierarchie.
   - Pre každý prvok $b_i \in B$ nájdi najbližšieho suseda $b'_i \in B$; a nech d_i je vzdialenosť b_i a b'_i.
   - Utrieď B vzhľadom na d_i.
   - Operáciu opakuj pre novovzniknuté uzly, ale pre väčšiu toleranciu pre d_i.

   Tato stratégia však produkuje „nepotrebné miesto“ vo vytvorených bunkách.



2. Druhý prístup je menej „greedy“.

Nech B je množina BV na najnižšej úrovni BV hierarchie a $|B| = n$.
- Algoritmus vypočíta stredy pre každý $b'_i \in B$;
- Utriedi B podľa osi x vzhľadom na c^i_x. Následne sa rozdelí B vertikálne (podľa c^i_x)
- Utriedi každý rez podľa osi y vzhľadom na c^i_y. Následne sa rozdelí rez vertikálne (podľa c^i_y)
-



Zhora – dole.
Najpoužívanejší prístup:

- Inicializácia je kompletná množina resp. jej maximálna BV
- Rozdelenie do k-častí.
- Vytvorenie BV stromu a rekurzívne delenie nových častí
-

Vloženie.
Dva algoritmy:

3.2 *Delenia priestoru*

- ***Pravidelná mriežka*** - Nie je hierarchiou, len rovnomerné rozdelenie priestoru
- ***Oktalovy strom*** - Delenie troma kolmými rezmi na súradnicové osi a umiestnené v polovici daného priestoru.
- ***Strom BSP*** – Binárny strom s ľubovoľne umiestnenými rezmi.
- ***kD-strom*** – špeciálny prípad BSP stromu, určený k popisu ľubovoľného k-rozmerného priestoru. Orezávajúce roviny sú kolmé na jednu z osí a orientácia rezov sa pravidelne strieda.

3.2.1 BSP Stromy

BSP strom – *Binary Space Partitioninig Tree*. Ide o binárny strom, v ktorom koreň stromu reprezentuje celý priestor scény. Každému vnútornému uzlu vrátane koreňa je priradená rovina. Táto rovina delí priestor prislúchajúci danému uzlu na dva menšie priestory, každý prislúchajúci inému z dvoch synov tohoto uzla. Z tejto základnej vlastnosti je odvodené aj meno celej štruktúry. Takéto delenie pokračuje rekurzívne na synoch. Delenie končí, ak uzol spĺňa dané kritéria. Tými môže byť hĺbka uzla, počet objektov ležiacich v tomto priestore, konvexnosť atď. Priestor prislúchajúci uzlu je prienik polpriestorov definovaných rovinami, ktoré sú priradené uzlom na ceste z tohoto uzla do koreňa BSP stromu. Celková efektivita a vhodnosť použitia záleží hlavne na výbere deliacich rovín v jednotlivých uzloch.

Najpoužívanejšie sú dva druhy rovín:

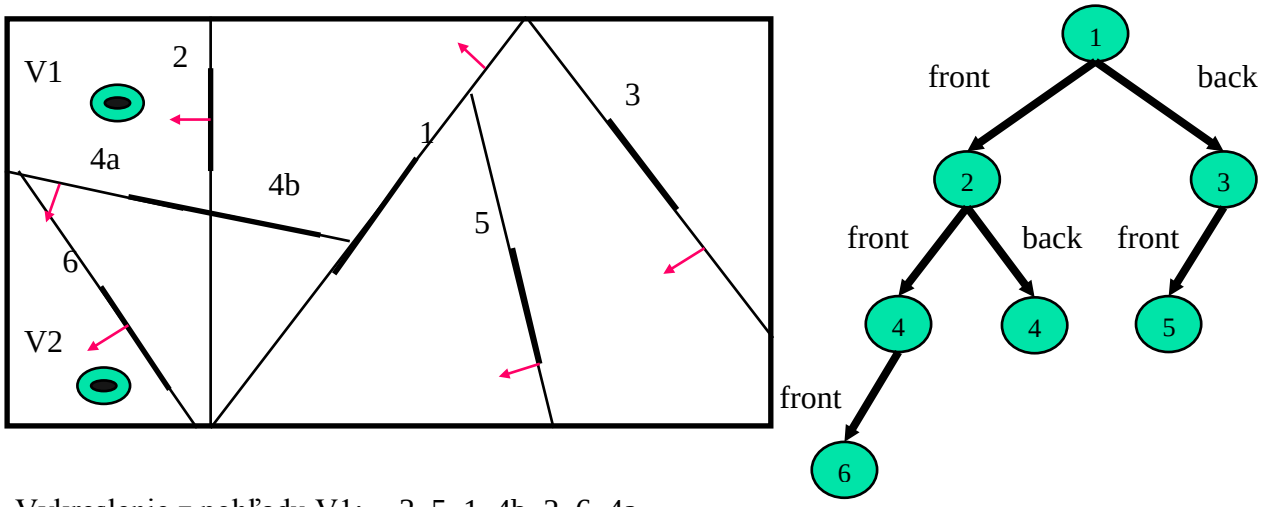
- roviny zarovnané ku osiam – vtedy sú telesá prislúchajúce jednotlivým uzlom kvádre so stenami zarovnanými ku osiam (anglicky Axis Aligned Box, skratka AAB)
- roviny zarovnané ku polygónom (pre 2D úsečkám)

Problém, s ktorým sa tu stretávame, je, že možných rovín spĺňajúcich predchádzajúce kritérium je veľa. Vybrať najvhodnejšiu rovinu však nie je jednoduché, nakoľko lokálne dobré riešenie nemusí viesť ku globálne dobrému celkovému riešeniu. Kritérií pre lokálnu vhodnosť môže byť veľa. Najčastejšie ide o vyváženosť oboch synov a počet objektov, ktoré táto rovina pretína. Objekt, s ktorým má rovina prienik, môžeme rozdeliť na 2 objekty tak, aby žiaden z nich táto rovina nepretínala a ich zjednotenie bol pôvodný objekt. Iné riešenie je tieto objekty uchovať v uzle (ak to algoritmy, ktoré budeme neskôr používať dovoľujú). Ďalšie riešenie je uchovať objekt (resp. jeho referenciu) duplicitne – v oboch synoch. Ktoré riešenie je najvhodnejšie záleží od konkrétnej aplikácie, pre ktorú strom konštruujeme

BSP strom: rozdelí cely priestor (preto partition) do binárneho stromu

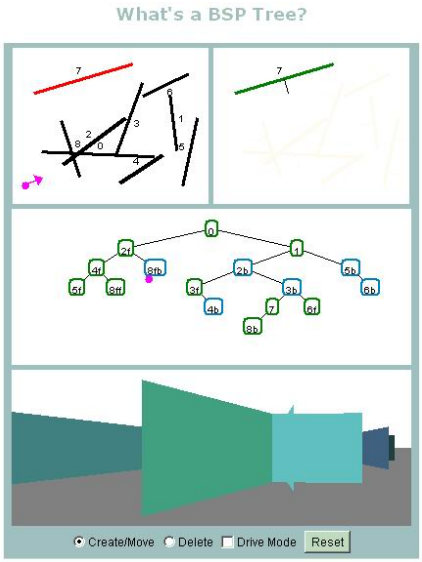
- Preprocess: vytvor binárny strom v scéne
- Runtime: korektne prechádzaj stromom od zozadu dopredu
- Idea: delenie priestoru rekurzivne do polpriestorov pomocou deliacich rovín (splitting planes)
    - Deliace roviny môžu byt ľubovoľne orientované

Príklad:



Vykreslenie z pohľadu V1: 3, 5, 1, 4b, 2, 6, 4a
Vykreslenie z pohľadu V2: 3, 5, 1, 4b, 2, 4a, 6

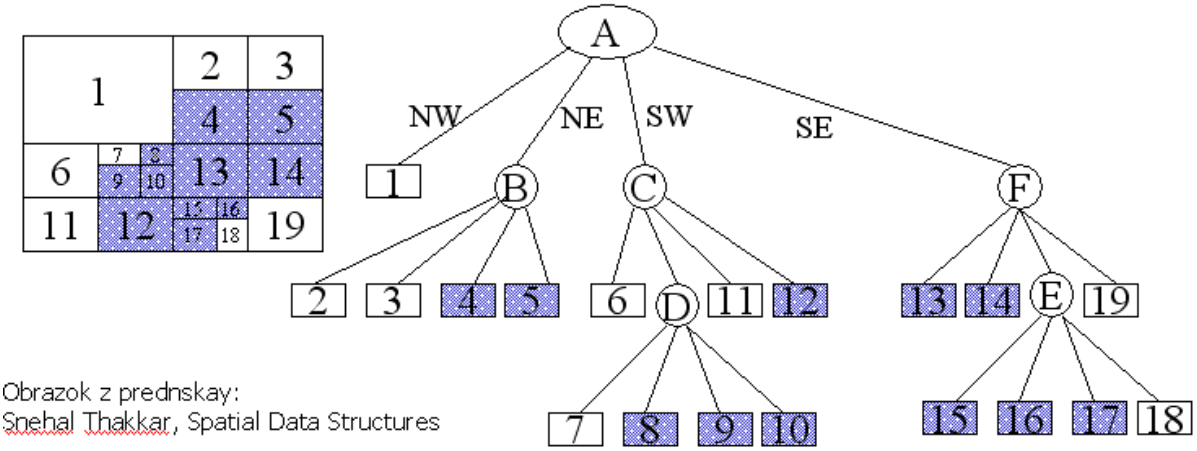
Demo: http://symbolcraft.com/graphics/bsp/index.html



Poznámka:
Varianta BSP stromu je ortogonálny BSP strom -> orezávajúce roviny sú kolmé na osi, no ich smer sa nemusí pravidelne striedať ako tomu bolo pri kD-stromoch.

3.2.2 Octree

Octree.
Delenie troma kolmymi rezmi, umiestnenými v polovici daneho priestoru. Delením vzniká osem oktantov



Obrazok z prednskay:
Snehal Thakkar, Spatial Data Structures

Octree je stromová štruktúra podobná BSP stromu s deliacimi rovinami zarovnanými ku osiam. Uzol reprezentuje ku osiam zarovnaný kváder – AAB. Tento kváder je rozdelený na osem rovnako veľkých kvádrov (deliaci bod je v strede) a každý z nich je uložený v inom z jeho ôsmych synov. Rovnaká veľkosť kvádrov vo všetkých synov daného uzla zabezpečuje, že všetky uzly na jednej úrovni majú rovnakú veľkosť.

Ďalšia vlastnosť z tohoto vyplývajúca je, že synovia majú prázdny prienik a zjednotenie ich kvádrov tvorí kváder ich otca. Teda zjednotenie všetkých uzlov na jednej kompletnej úrovni pokrýva celý priestor prislúchajúci koreňu stromu. Teda octree je regulárna štruktúra, čo môžu algoritmy efektívne využiť. Najčastejšie je kváder koreňa najmenší AAB ohraničujúci celú scénu.

Kritériá pre ukončenie vetvenia uzla na synov sú podobné ako v prípade BSP stromu. Ak sa používa AAB uložené v uzle súčasne aj ako ohraničujúci objekt (presnejšie AABB), tak taktiež problémy s objektmi, ktoré sú na hraniciach uzlov, sa riešia podobne ako pri BSP strome (delením, duplikovaním alebo uložením v najspodnejšom uzle, do ktorého AABB sa objekt kompletne zmestí). Dvojrozmerná verzia octree štruktúry sa nazýva quadtree.

Problémy s objektmi ležiacimi na hraniciach AABB v octree rieši štruktúra **loose-octree**. Základ je taký istý ako v pôvodnom octree, teda každý uzol má svoj stred, ktorý je stredom AAB tohoto uzla. Princíp spočíva v tom, že okolo tohoto stredného bodu uzla je definovaný ešte jeden AAB s k-krát väčšou stranou ako má pôvodný AAB. Ohraničujúcim telesom nie je v tomto prípade pôvodný AAB, ale nový – k-krát väčší. Týmto sa môže podstatne znížiť počet objektov, ktoré pretínajú AABB daného uzla a teda môžu byť uložené nižšie v hierarchii (prípadne byť neskôr delené alebo mať menej duplikátov).

Zaujímavú vlastnosť má štruktúra pre k=2. Vtedy vieme objekt veľkosti menšej ako polovica veľkosti uzla určite vložiť do jedného zo synov tak, aby objekt nepretínal AABB tohoto syna. To, do ktorého syna objekt vložíme, záleží od stredu tohoto objektu – pôjde do toho syna, ktorého pôvodný nezväčšený AAB obsahuje stred AABB vkladaného objektu. Takýto syn existuje práve jeden. Takáto štruktúra je veľmi dobrá pre spracovanie dynamických objektov, nakoľko každý objekt je v štruktúre uložený práve raz a existuje práve jedna pozícia v štruktúre, kde môže byť uložený. Naviac je uložený v uzle s dostatočne malým AABB. Pri loose-octree však strácame niektoré vlastnosti, napríklad možnosť presne zoradiť uzly v danom smere, nakoľko sa prekrývajú svojimi AABB.

3.2.3 R-tree

R-tree – R-strom je stromová štruktúra podobná B-stromom. Je optimalizovaná na veľkosť blokov na disku a stránok v pamäti. Ako ohraničujúce telesá používa AABB. Podobne ako B-strom majú vnútorné uzly ohraničený počet synov. Tento strom je vyvážený a teda nie je zbytočne vysoký. AABB synov daného uzla môžu mať neprázdny prienik.

Problémom pri tejto štruktúre je vložiť, zmazať i posunúť objekt. Vždy treba robiť rozhodnutie, do ktorého syna sa najviac oplatí objekt vložiť (je pravdepodobné, že tomuto uzlu sa bude zväčšovať jeho AABB). V prípade pohybu objektu sa zasa zisťuje, či sa neoplatí presunúť objekt do iného uzla miesto zväčšenia AABB. Presunutie do iného uzla môže znamenať aj viacero zmien v štruktúre, aby si zachovala svoje vlastnosti. Pohyb objektov sa často rieši zmazaním a opätovným vložením objektu do hierarchie (nie len v prípade R-stromov). Teda v závislosti od rozhodnutia, ktorý syn sa zväčší alebo či sa vytvorí nový syn a podobne, závisí efektivita celej štruktúry. Implementácia takejto štruktúry je podstatne zložitejšia ako v prípade predchádzajúcich štruktúr (BSP-tree, octree, quadtree, resp. ich loose verzie).

Predchádzajúce štruktúry mali vlastnosť, že sme vedeli pomocou takejto štruktúry veľmi rýchlo zoradiť objekty v danom smere. V prípade loose verzií šlo iba o približné usporiadanie. V prípade R-stromov je táto vlastnosť ešte viac porušená, nakoľko poloha AABB synov uzla v priestore je takmer náhodná a môžu sa takmer ľubovoľne prekrývať. Ďalšia z nevýhod R-stromov, spôsobená možným prekrývaním uzlov, je nejednoznačnosť, v ktorom uzle sa nachádza konkrétny objekt. To robí problém napríklad pri odstraňovaní objektu zo štruktúry.

3.2.4 kD-tree

Špeciálny prípad BSP stromu, určený k popisu ľubovoľného k-rozmerného priestoru. Orezávajúce roviny sú kolmé na jednu z osí a orientácia rezov sa pravidelne strieda.

3.2.5 Zhrnutie

Niektoré algoritmy viditeľnosti pracujú tak, že zistia viditeľnosť všetkých buniek nejakého uniformného rozdelenia scény. Najčastejšie je scéna rozdelená uniformnou mriežkou. Potom zistiť viditeľnosť AABB podľa viditeľnosti jednotlivých políčok uniformnej mriežky je tiež jednoduchší proces ako v prípade OBB či k-DOP. Naviac je to presnejšie, nakoľko tieto uniformné mriežky vytvárajú najčastejšie bunky tvaru AAB.

Niektoré algoritmy zisťujú viditeľnosť v obrazovom priestore. Guľa sa po projekcii do tohoto priestoru stáva akousi elipsou a úloha zistiť, ktoré pixle patria do tejto elipsy je časovo náročná. Naviac pre objekty virtuálneho mesta ako sú budovy a terén je ohraničenie pomocou AABB omnoho „tesnejšie“ ako ohraničenie guľou. Teda objem AABB je menší ako objem gule. V prípade gule je teda väčšia pravdepodobnosť, že neviditeľný objekt bude algoritmom označený ako aspoň čiastočne viditeľný. Teda AABB je lepší ako guľa v tomto ohľade. Taktiež projekcia k-DOPu je podstatne náročnejšia ako projekcia AABB, nakoľko má viac vrcholov a stien.

V dynamických scénach sa množstvo objektov pohybuje. S narastajúcim počtom pohybujúcich sa objektov je nutné neustále aktualizovať stále väčšiu časť hierarchie ohraničujúcich telies. Toto môže viesť ku problémom v nedostačujúcim výkonom. Ako riešenie sa dajú použiť viaceré prístupy. Jedným z riešení je pohybujúci objekt vyňať z hierarchie na dobu jeho pohybu.

Iným riešením je využiť vlastnosti jeho pohybu. V praxi sa objekty pohybujú po spojitej krivke. Teda z aktuálnej rýchlosti, smeru pohybu a rotácie dokážeme odhadnúť, kde sa objekt bude nachádzať v niekoľko po sebe idúcich krokoch. Teda riešenie je zmeniť ohraničujúce teleso na také, ktoré ohraničuje tento objekt na všetkých predpokladaných pozíciách v blízkej budúcnosti. Takéto ohraničujúce teleso pohybujúceho sa objektu sa bude meniť znova až keď teleso ukončí svoj pohyb alebo v prípade, že sa teleso ocitne mimo jeho ohraničujúceho telesa. Takýmto prístupom sa podstatne zmenší rozsah aktualizovania hierarchie v jednom kroku.

V závislosti od veľkosti a zložitosti telesa je vhodný prvý alebo druhý prístup. Pri prístupe s ohraničujúcimi telesami platnými niekoľko po sebe idúcich krokov je možné v prípade označenia objektu ako viditeľného spraviť naviac test viditeľnosti na jeho tesné ohraničujúce teleso. Teda objekt by mal dve ohraničujúce telesá, jedno tesne ohraničujúce a druhé voľné (s rezervou) s ohľadom na charakter jeho pohybu. To má význam hlavne u rýchlo sa pohybujúcich telies, kedy je dočasné ohraničujúce teleso podstatne väčšie ako pôvodné tesne ohraničujúce teleso. Teda tesné teleso by sa muselo spočítať iba v prípade, ak by bolo jeho voľné ohraničujúce teleso viditeľné.

4 Literatura:

[1] Benes Bedřich, Felkl Petr, Sochor Jiří, ?ára Jiří. *Moderni pocitacova grafika*, 2. vydanie, Computer Press, 2004, ISBN: 80-251-0454-0.

[2] Gabriel Zachmann, Elmar Langetepe. *Geometric Data Structures for Computer Graphics*, Siggraph 2003 Tutorial, San Diego on Monday, July 28.

[3] Tomas Akenine-Möller, Eric Haines. *Real-time rendering, 2nd edition*, A. K. Peters Limited. 2002, ISBN 1-56881-182-9.

[4] Daniel Cohen-Or, Yiorgos Chrysanthou, Claudio T. Silva, Fredo Durand. *A Survey of Visibility for Walkthrough Applications*. SIGGRAPH 2001.