Diskrétne geometrické štruktúry

6.

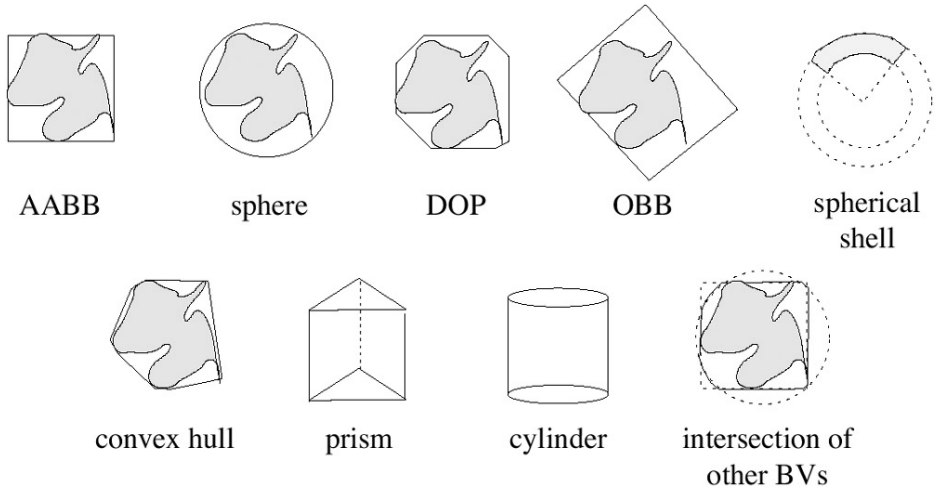
Martin Florek

florek@sccg.sk

www.sccg.sk/~florek

Ohraničujúce objemy

- pomocná štruktúra ohraničujúca jeden alebo viac objektov
- jednoduchý tvar objemu
- nevhodné na reprezentáciu
- prenos výpočtu z objektu na objem



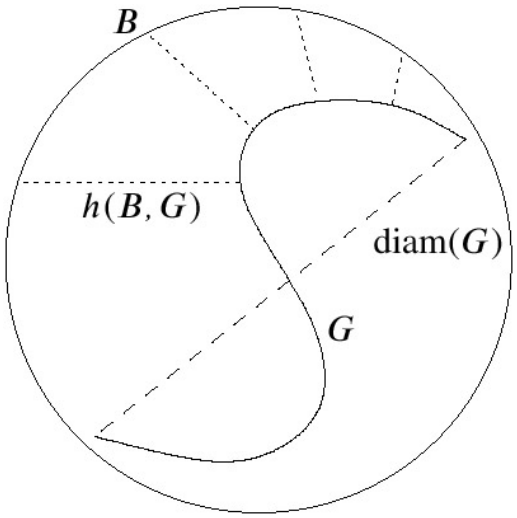
Presnosť aproximácie

- ohodnotenie blízkosti ohraničujúceho objemu k objektu
- použitie Haussdorfovej vzdialenosti
- záleží aj na veľkosti objemu

$$h(B,G) = \max_{b \in B} \min_{g \in G} d(b,g)$$

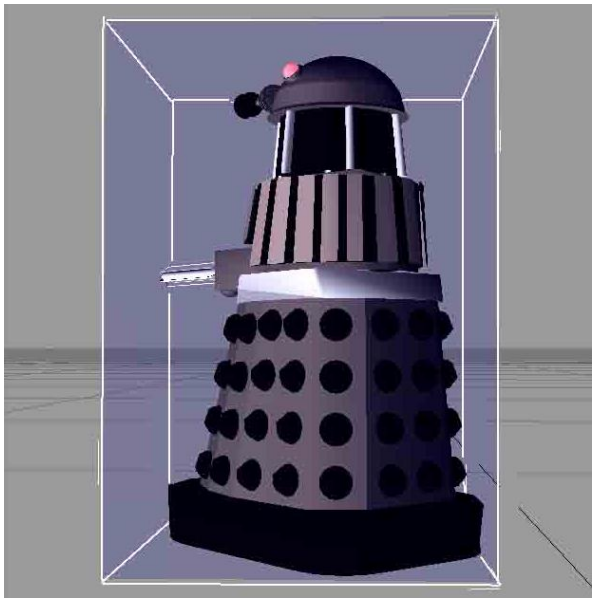
$$\operatorname{diam}(G) = \max_{g,f \in G} d(g,f)$$

$$\tau := \frac{h(B,G)}{\operatorname{diam}(G)}.$$



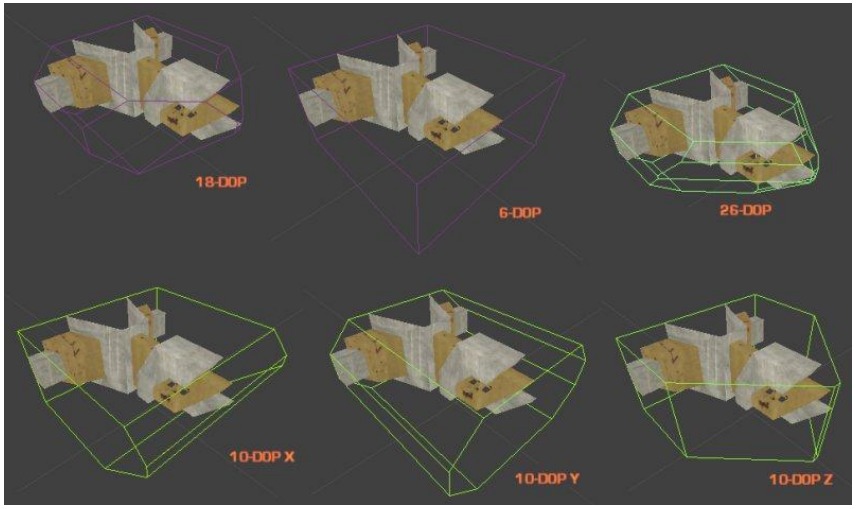
AABB

- axis-aligned bounding box
- jednoduché vybudovanie objemu
- jednoduché prieniky
- horšia aproximácia, až 0,5



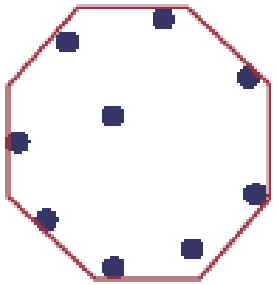
k-DOP

- Discrete Oriented Polytope
- polytop – polygon, polyhedron...
- zovšeobecnenie AABB
- prienik *k* polpriestorov
- konvexný obal je najmenší k-DOP

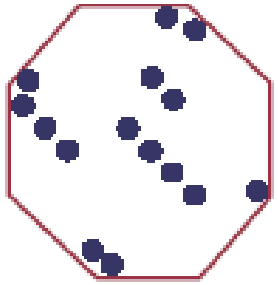


k-DOP 2

- lepšia aproximácia, záleží na objekte
- 2D: k = 4, 8; 3D: k = 6, 18, 26
- podobné zložitosti algoritmov ako pre AABB



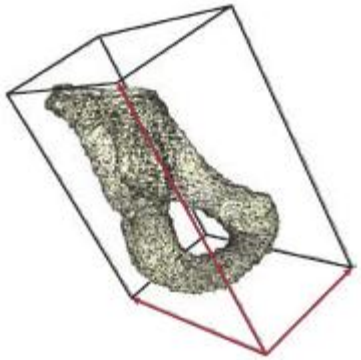
good choice



quite good choice

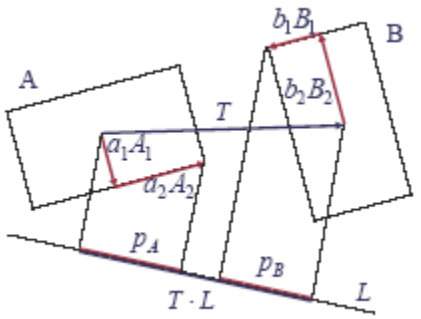
OBB

- Oriented Bounding Box
- ľubovoľná orientácia
- presnejšia aproximácia
- časovo náročnejšie testy
- najčastejší výpočet – orientácia



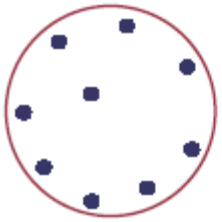
OBB 2

- prienik OBB – najdenie „separating axis“
  - priamka na ktorej sa projekcie polytopov nepretínajú
- dva objekty A a B sa nepretínajú ak existuje priamka *p* taká, že priemety A a B na priamku *p* sa nepretínajú
- smer *p* je definovaný orientáciou stien A a B a vektorovým súčinom hrán z A a B

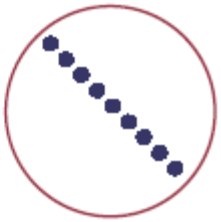


Sféra

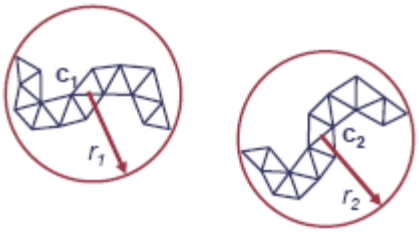
- vhodná pre transformácie
- ľahké výpočty
- vytvorenie: najmenšia ohraničujúca kružnica, kružnica ohraničujúca AABB, OBB, ...



good choice



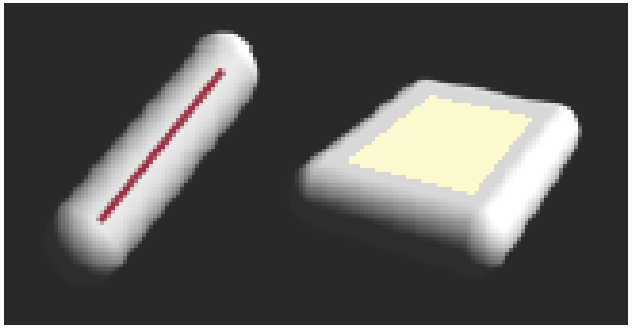
bad choice

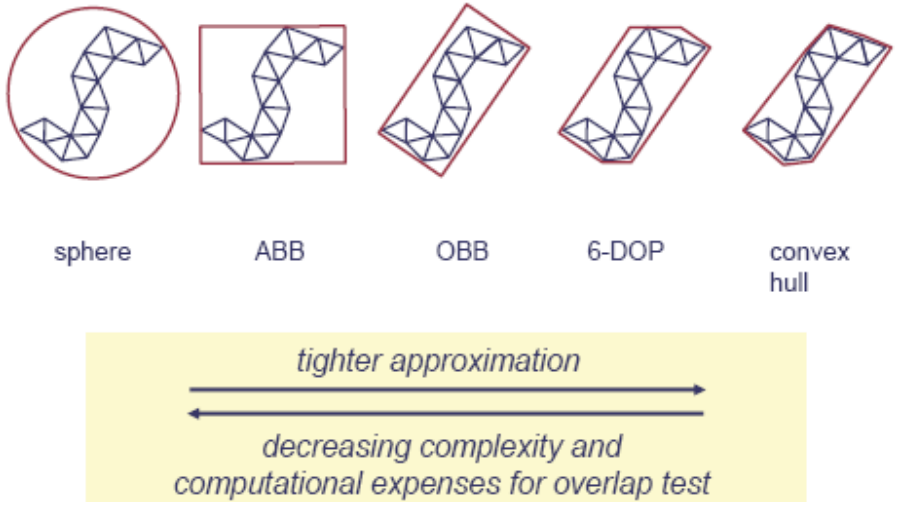


$$(\mathbf{c}_1-\mathbf{c}_2)(\mathbf{c}_1-\mathbf{c}_2) > (r_1+r_2)^2$$

Ďalšie objemy

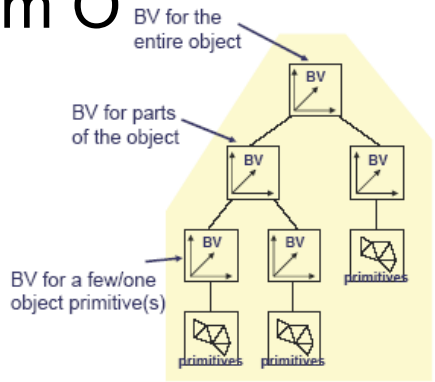
- elipsoidy
- cylindre, hranoly
- konvexné obaly
- rozšírené sférické objemy
- kombinácie



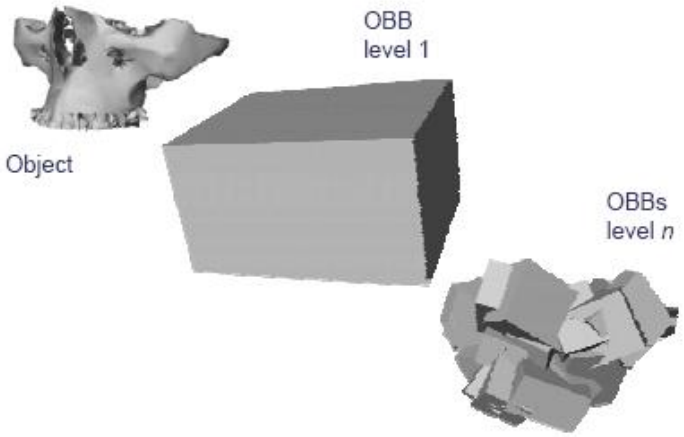


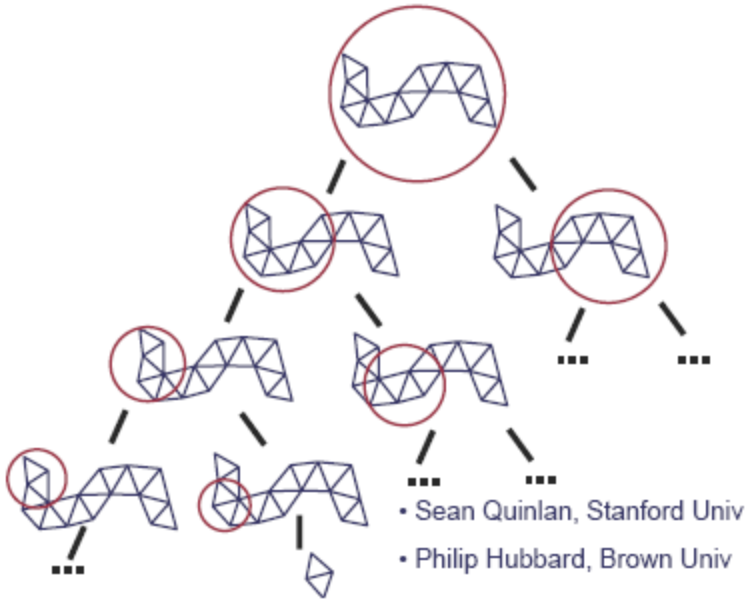
Bounding Volumes Hierarchy

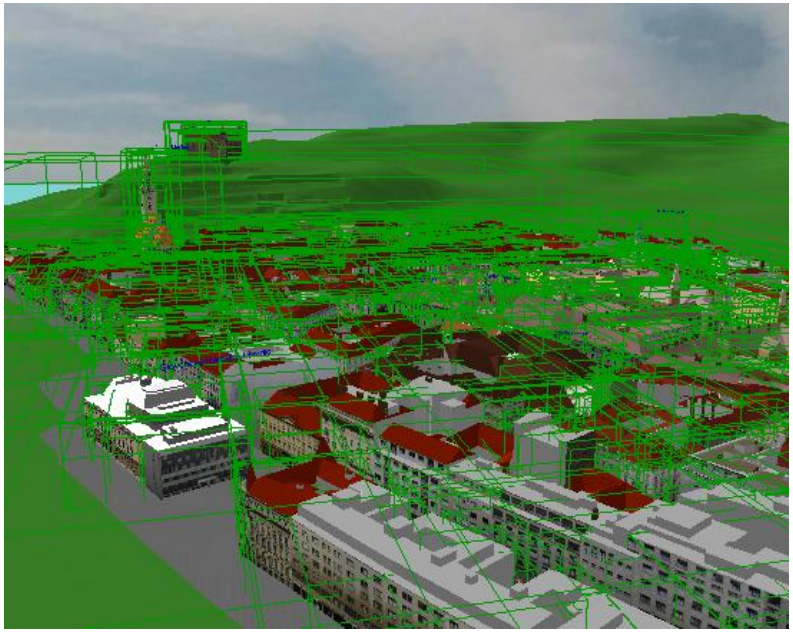
- strom ohraničujúcich objektov
- nech $O = \{o_1, \ldots, o_n\}$ je množina objektov
- potom je BVH pre množinu O definovaný:
  - ak $|O| \leq e$, tak BVH(O) je list ktorý obsahuje O a ohraničujúci objem O
  - ak $|O| > e$, tak BVH(O) je vrchol s n potomkami v_i, i=1,...,n, kde v_i sú BVH(O_i) nad množinami O_i, $O = \cup O_i$, v obsahuje aj ohraničujúci objem O



Príklady


OBB
level 1
Object
OBBs
level *n*


• Sean Quinlan, Stanford Univ
• Philip Hubbard, Brown Univ

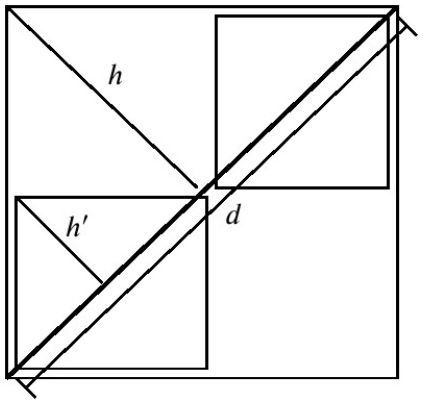


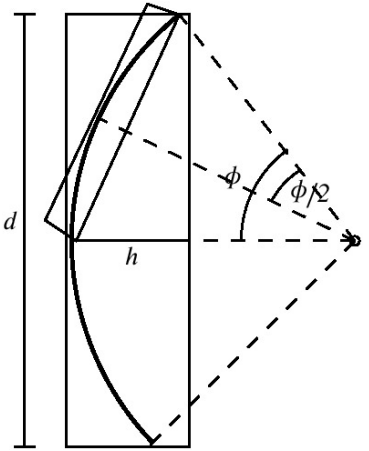
Tesnosť aproximácie BVH

- volumetrická tesnosť
- C(v) potomkovia *v*
- L(v) listy *v*
- pri AABB sa tesnosť nemení, závisí na orientácii geometrie
- pri OBB sa blíži k 0, závisí na krivosti

$$\tau := \frac{\mathrm{Vol}(v)}{\sum_{v' \in C(v)} \mathrm{Vol}(v')}.$$

$$\tau := \frac{\mathrm{Vol}(v)}{\sum_{v' \in L(v)} \mathrm{Vol}(v')},$$





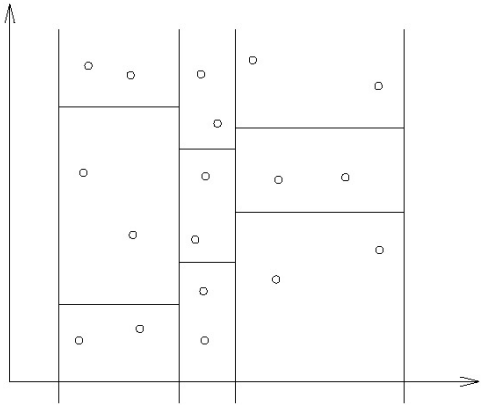
Konštrukcia

- Top-down
  - začneme s celou scénou, objektom
  - vytvoríme ohraničujúci objem pre scénu
  - objem (objekt) rozdelíme na *k* častí
  - prerozdelujeme
  - skončíme keď objem obsahuje podprahový počet primitív

Konštrukcia 2

- Bottom-up:
  - na začiatku primitívy reprezentujúce objekt
  - vytvor ohraničujúci objem pre každú primitívu
  - rekurzívne zoskupuj ohraničujúce objemy do väčších celkov
  - ukončenie keď zostane jeden veľký objem

Spôsoby zoskupovania



- Greedy:
  - usporiadaj objemy podľa vzdialeností od seba
  - v usporiadaní vyber prvých *k* objemov a vytvor pre ne nového predka v hierarchii
- Tiling:
  - pre každý objem vypočítaj stred
  - rozdeľ priestor na $\sqrt[d]{n/k}$ rezov tak, aby v každom bol rovnaký počet primitív (podľa *x* a potom podľa *y*)
  - vznikne rozdelenie priestoru na *k* dielov, objemy v jednej bunke spojíme do jedného objemu
  - pokračujeme o úroveň vyššie

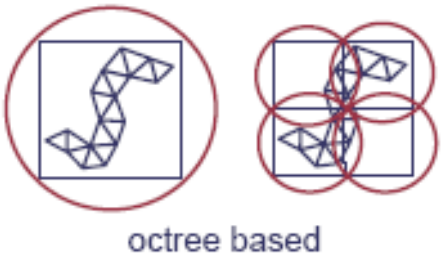
Konštrukcia 3

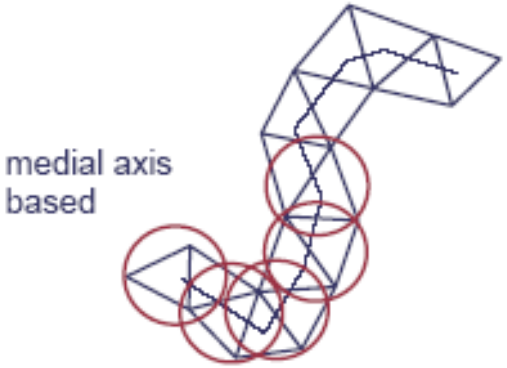
- vkladaním

```
void BVH_Insert (B)
{
        while (|B| > 0)
        {
                b = B.pop;  v = root;   // vybratie dobreho b je dolezite
                while (!v.IsLeaf())
                {
                        choose child v′, so that insertion of b into v′ causes minimal increase
                        in the costs of the total tree;
                        v = v′;
                }
        }
}
```

Konštrukcia Sférických BVH

- Hubbard, C. O'Sullivan:
  - approximuj primitívy pomocou sfér a vytvor strom bottom-up zoskupovaním sfér
  - pri tvorbe vytváraj homogénnu štruktúru vrcholov pre odstránenie redundancie
  - vytvor top-down BVH pomocou octree štruktúry
  - vypočítaj strednú os a polož stredy sfér sem





Konštrukčné kritériá

- vyvážený strom
- delenie objektov alebo objemov
- ako deliť objekty, objemy
- minimalizácia redundancie (či sa objekt nachádza vo viacerých vrcholoch)
- počet primitív v listoch
- podľa použitia stromu
- ohodnotenie vrcholov

Ohodnotenia

- raytracing: ohodnocovacia funkcia na základe plochy, α = x, y, z, ...

$$k^{\alpha} = \min_{j=0\ldots n} \left\{ \frac{\text{Area}(b_1, \ldots, b_j)}{\text{Area}(B)} j \; + \frac{\text{Area}(b_{j+1}, \ldots, b_n)}{\text{Area}(B)} (n - j) \right\}$$

- frustum culling: objem namiesto povrchu
- collision detection: porovnávanie viacerých stromov a častí stromov, minimalizovať C(A, B)

$$C(A, B) = 4 + \sum_{i,j=1,2} P(A_i, B_j) \cdot C(A_i, B_j),$$

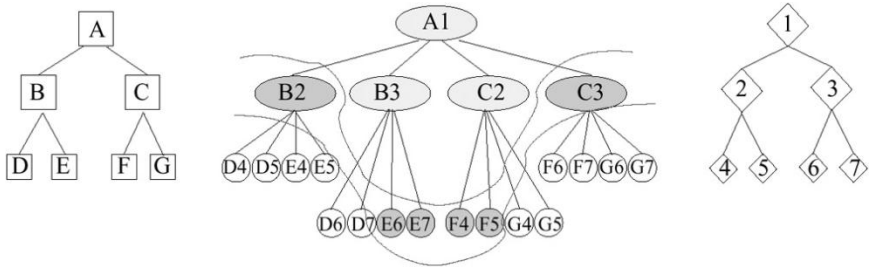
$$C(A, B) \approx 4\big(1 + P(A_1, B_1) + \ldots + P(A_2, B_2)\big).$$

$$P(A_1, B_1) \approx \frac{\text{Vol}(A_1) + \text{Vol}(B_1)}{\text{Vol}(A) + \text{Vol}(B)}.$$

Detekcia kolízií

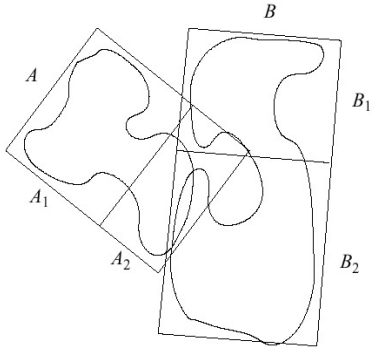
- kontrola či sa nejaké objekty pretínaju → kontrola prieniku ich primitív
- prechod dvoma stromami
- vytváranie bounding volume test tree (BVTT)

```
traverse(A,B)
{
        if (A ∩ B == 0)
                return NULL;
        if (A.IsLeaf() && B.IsLeaf())
                return A.primitives ∩ B.primitives;
        for (all children A[i] and B[j])
                traverse(A[i],B[j]);
}
```



Zlepšovanie detekcie

- skutočná kolízia iba v častiach objektu
- uchovávanie hladín na ktorých sa prechod zastaví
- pri malej transformácii objektu sa hladina zmení len málo - incremental hierarchical collision detection



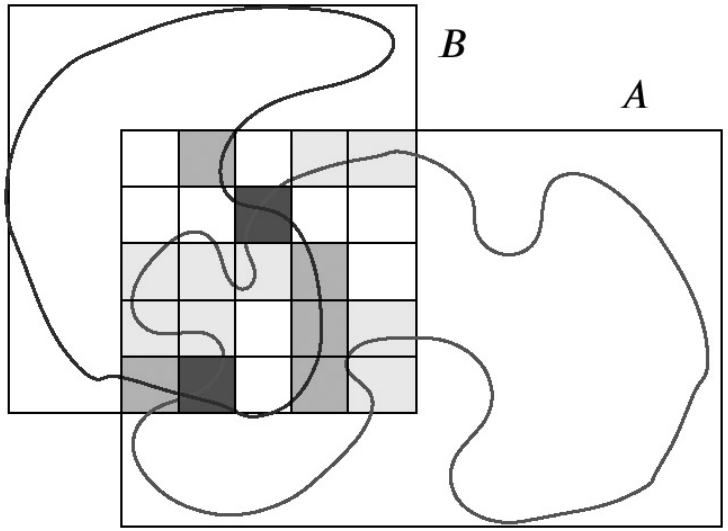
Stochastická detekcia kolízií

- ohodnotíme časti stromu, ktoré sa budú spracuvávať s vyššou prioritou
- pravdepodobnosť prieniku → externý opis správania sa primitív v objeme

```
traverse(A, B)
{
        q.insert(A, B);
        while (!q.IsEmpty())
        {
                (A, B) = q.pop();
                for (all children A_i and B_j)
                {
                        p = P [A_i, B_j];
                        if (p > thresh)
                                return “collision”;
                        else if (p > 0)
                                q.insert(A_i, B_j, p);
                }
        }
}
```

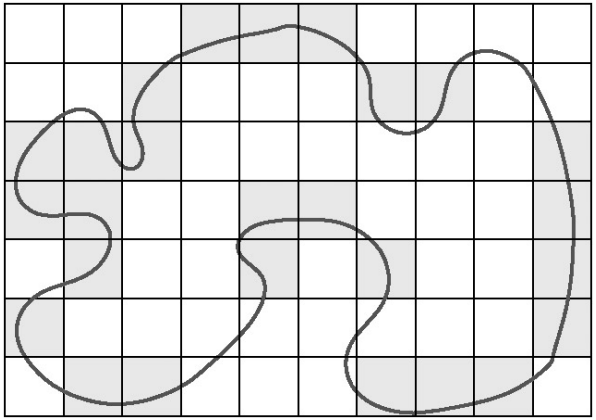
Pravdepodobnosť prieniku

- rozdelíme A∩B na regulárnu mriežku
- v každej bunke určíme obsadenosť bunky množinou A a B
- spočítaj počet kolíznych buniek



Stochastické riešenie

- v čase tvorby ohraničujúceho objemu
- rozdelíme objem regulárnou mriežkou
- zrátame počet dostatočne obsadených buniek a uložíme tento počet s objemom - s_A



$$P[c(A \cap B) \geq x] = 1 - \sum_{t=0}^{x-1} \frac{\binom{s_A}{t}\binom{s - s_A}{s_B - t}}{\binom{s}{s_B}}.$$

Graf scény

- založený na hierarchiách v scéne
- rýchlejší prístup k dátam
- môže byť kombinovaný s ohraničujúcimi objemami
- dynamické scény: bottom-up update
- každý vrchol má mnoho atribútov:
  - lokálna, globálna transformácia
  - renderovacie atribúty
  - ohraničujúci objem

koniec (-:

florek@sccg.sk