

Reprezentace 3D scén

© 1995-2019 Josef Pelikán
CGG MFF UK Praha

pepca@cgg.mff.cuni.cz
<https://cgg.mff.cuni.cz/~pepca/>



Metody reprezentace 3D scén

Objemové reprezentace

- přímé informace o vnitřních objemech těles
- snadný test „**bodxtěleso**“ (leží daný bod uvnitř tělesa?)
- **zobrazování** může být obtížnější
- používají se též jako **pomocné datové struktury** pro rychlé vyhledávání

Povrchové reprezentace

- přímé informace o povrchu těles (hrany, stěny)
- obtížnější test „**bodxtěleso**“ (tělesa vůbec nemusí mít vnitřní objem)
- poměrně snadné **zobrazování**



Objemové reprezentace

Výčtové reprezentace

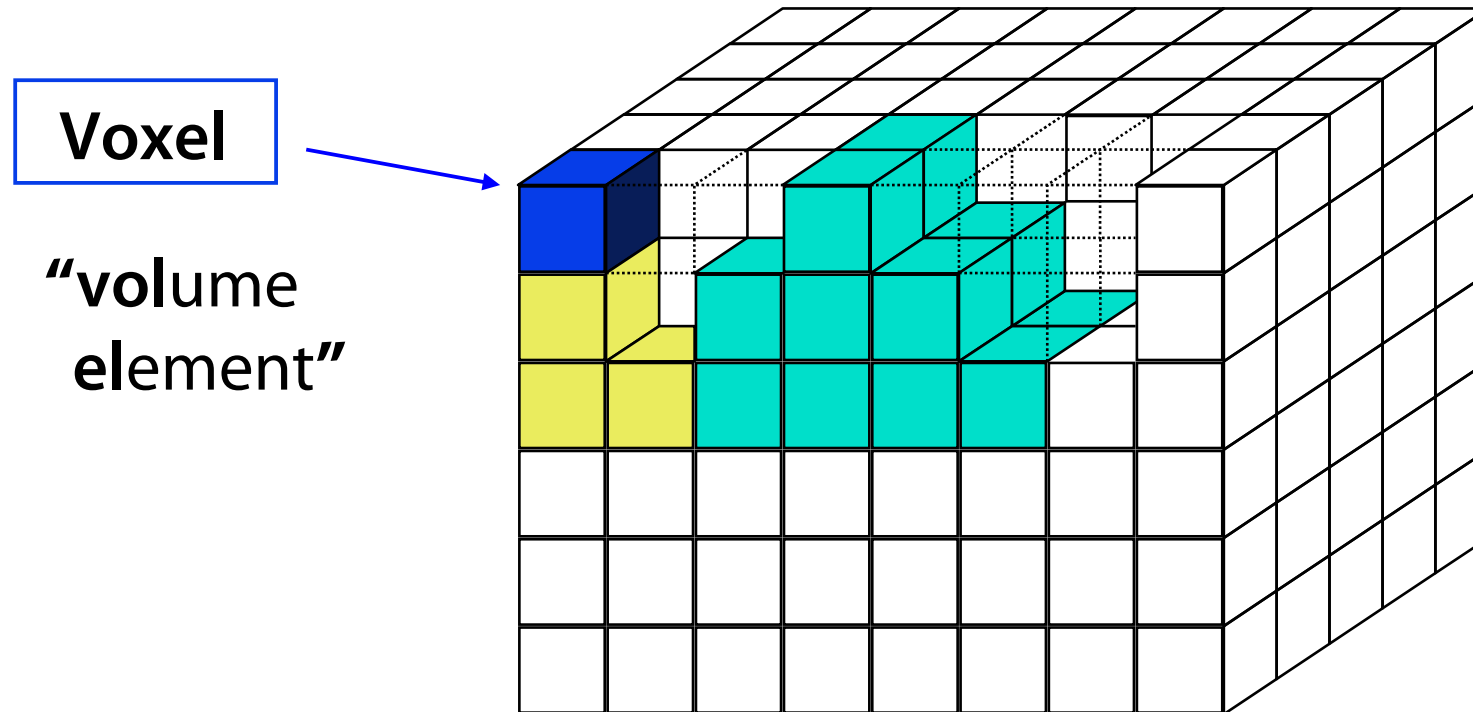
- přímé vyčíslení obsazeného prostoru (diskrétní reprezentace – omezená přesnost)
- používají se hlavně jako pomocné datové struktury pro rychlé vyhledávání
- **buněčný model, oktantový strom**

CSG (Constructive Solid Geometry)

- velice silná a přesná metoda (elementární tělesa, geometrické transformace, **množinové operace**)
- obtížnější **zobrazování** (vrhání paprsku)



Buněčný model



Pole $k \times l \times m$ voxelů

Jednabitová varianta: 0 – nic, 1 – těleso

Vícebitová varianta: 0 – nic, $n > 0$ – těleso číslo n



Zobrazování buněčného modelu

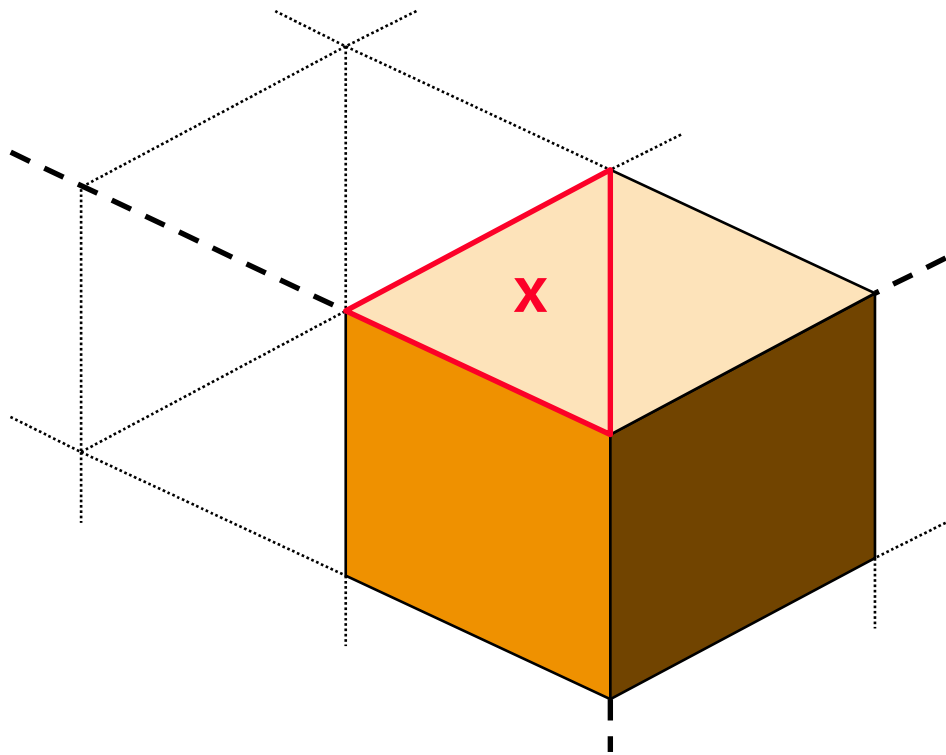
Kreslení odzadu-dopředu

- pouze přivrácené stěny voxelů
- pouze stěny na povrchu těles (stěny mezi 0 a >0)
- vícenásobné překreslování

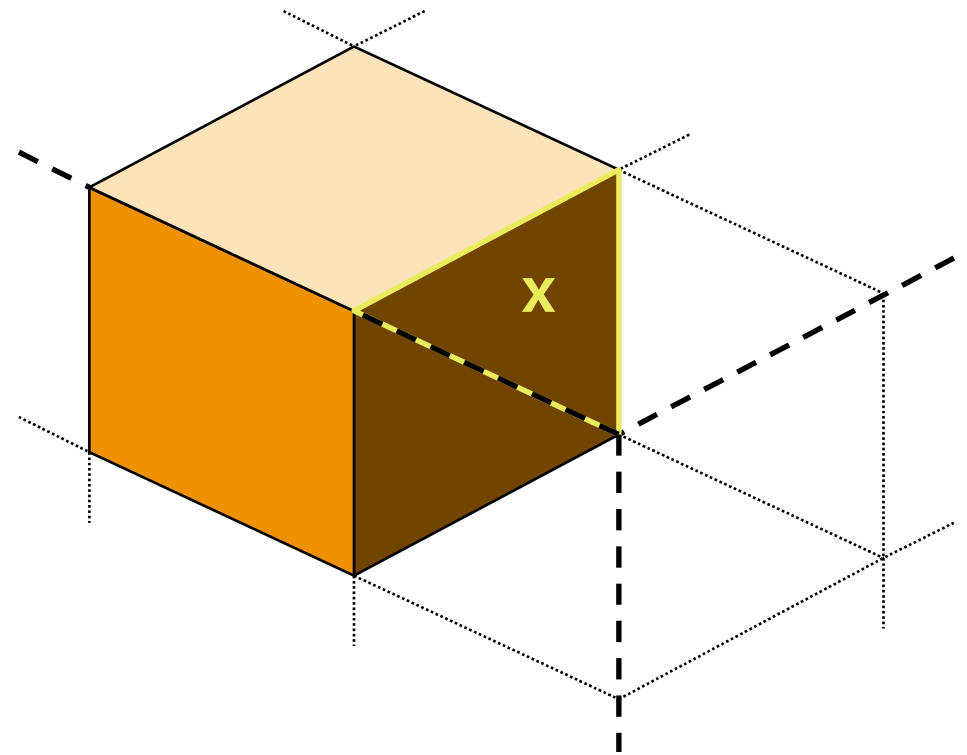
Speciální promítání

- velmi efektivní algoritmus bez zbytečného překreslování
- „**Ant-attack**“ na ZX-Spectru (128×128×8 voxelů)

Speciální promítání

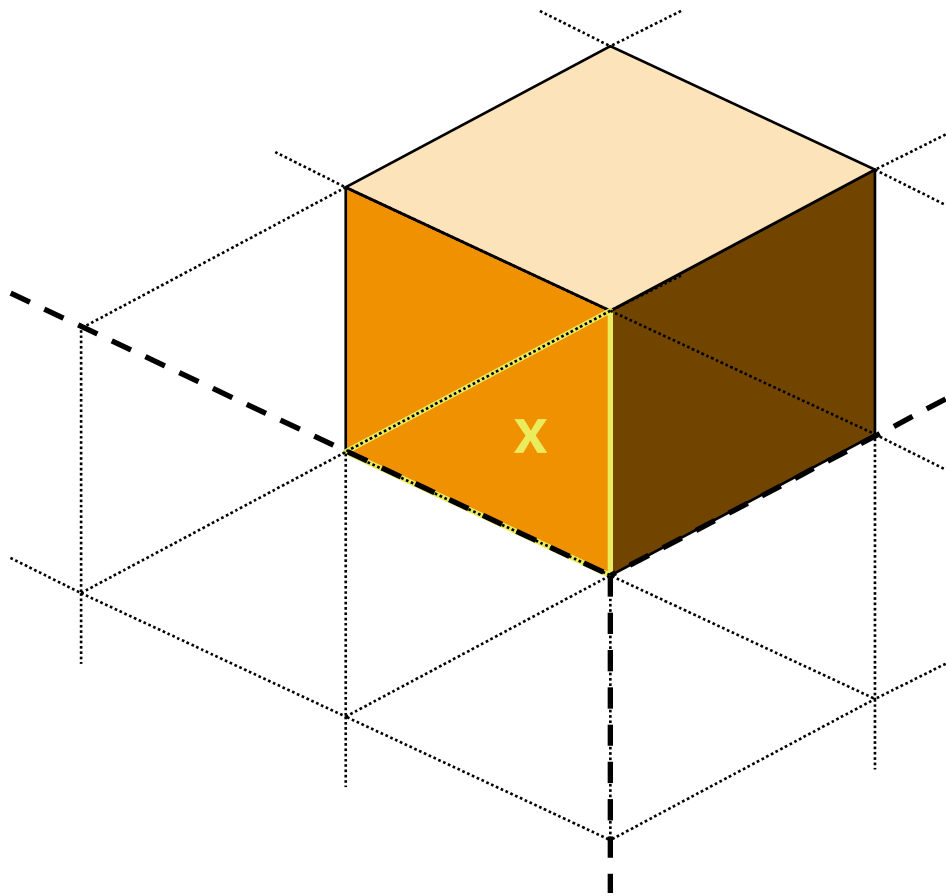


1. horní stěna
[0,0,0]

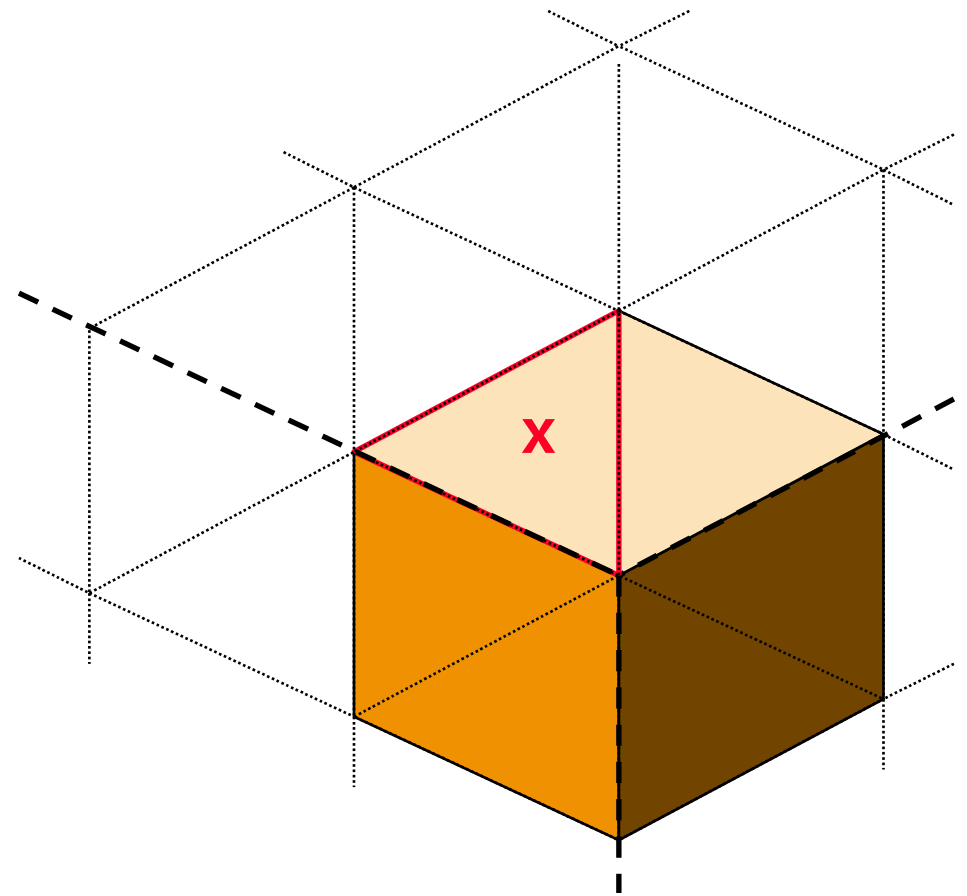


2. pravá stěna
[0,1,0]

Speciální promítání II

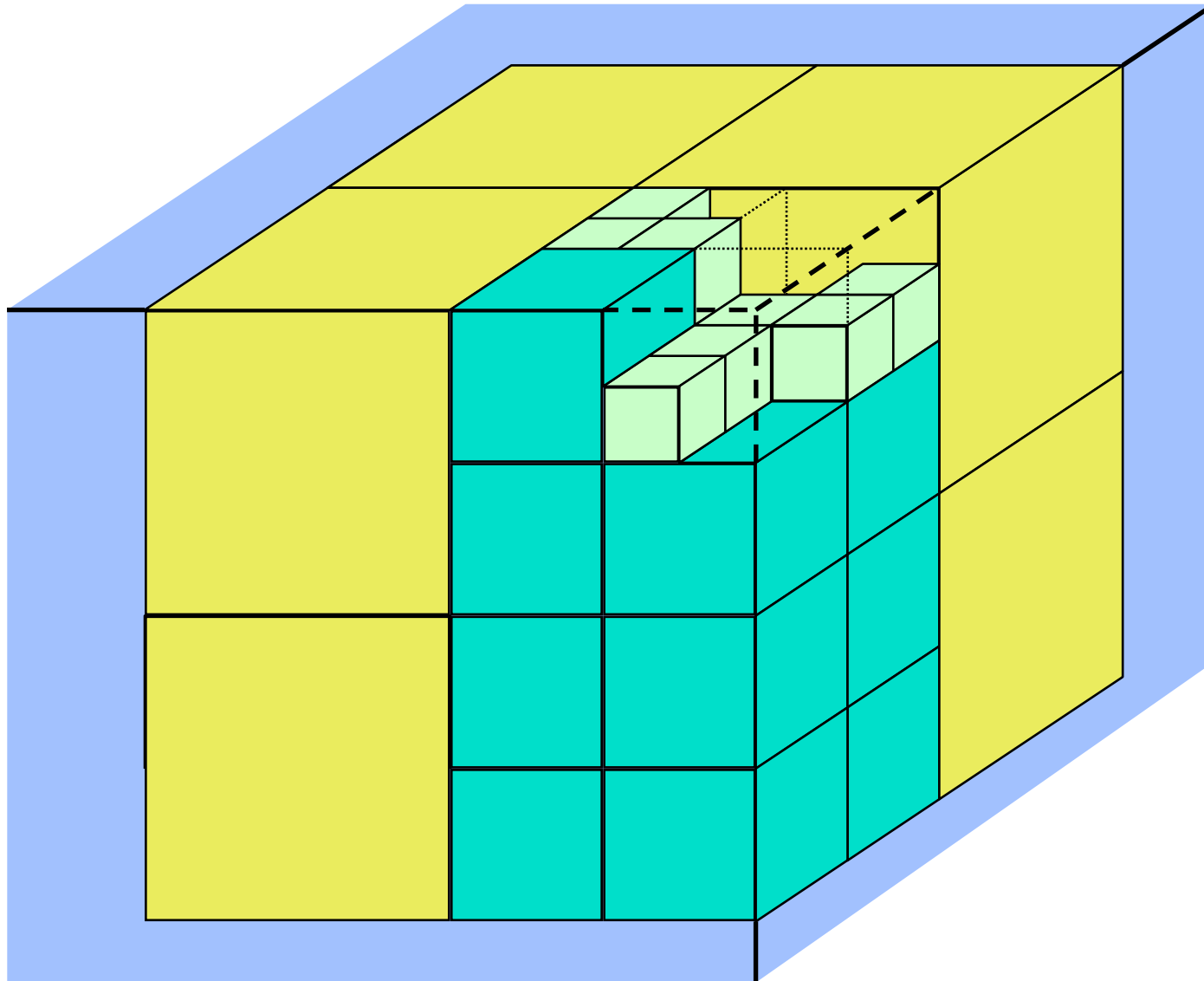


3. levá stěna
 $[1,1,0]$



4. horní stěna
 $[1,1,1]$

Oktantový strom („Octree“)





Oktantový strom

3D analogie kvadrantového stromu

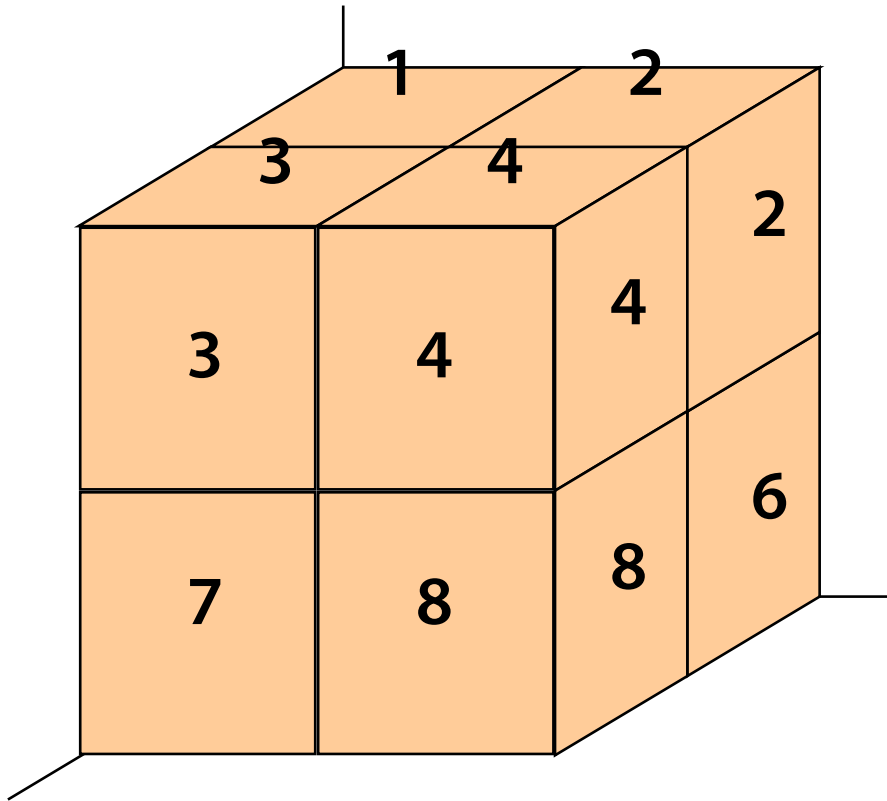
- je-li vnitřek krychle nehomogenní, rozdělí se na osm částí (dělí se až do úrovně voxelu)
- úspora paměti proti buněčnému modelu

Kreslení odzadu-dopředu

- pouze přivrácené stěny krychlí
- pouze stěny na povrchu těles (stěny mezi 0 a >0)
- několikanásobné překreslování některých pixelů

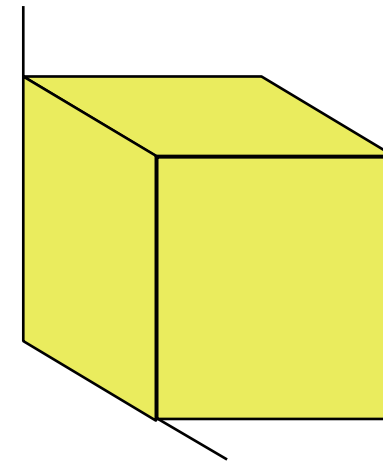


Kreslení odzadu-dopředu



Pořadí

5-6-1-2-7-8-3-4



Pořadí

6-5-2-1-8-7-4-3

CSG (Constructive Solid Geometry)



Elementární geometrická tělesa

- snadno definovatelná a vyčíslitelná
- kvádr, poloprostor, hranol, koule, válec, kužel...

Množinové operace

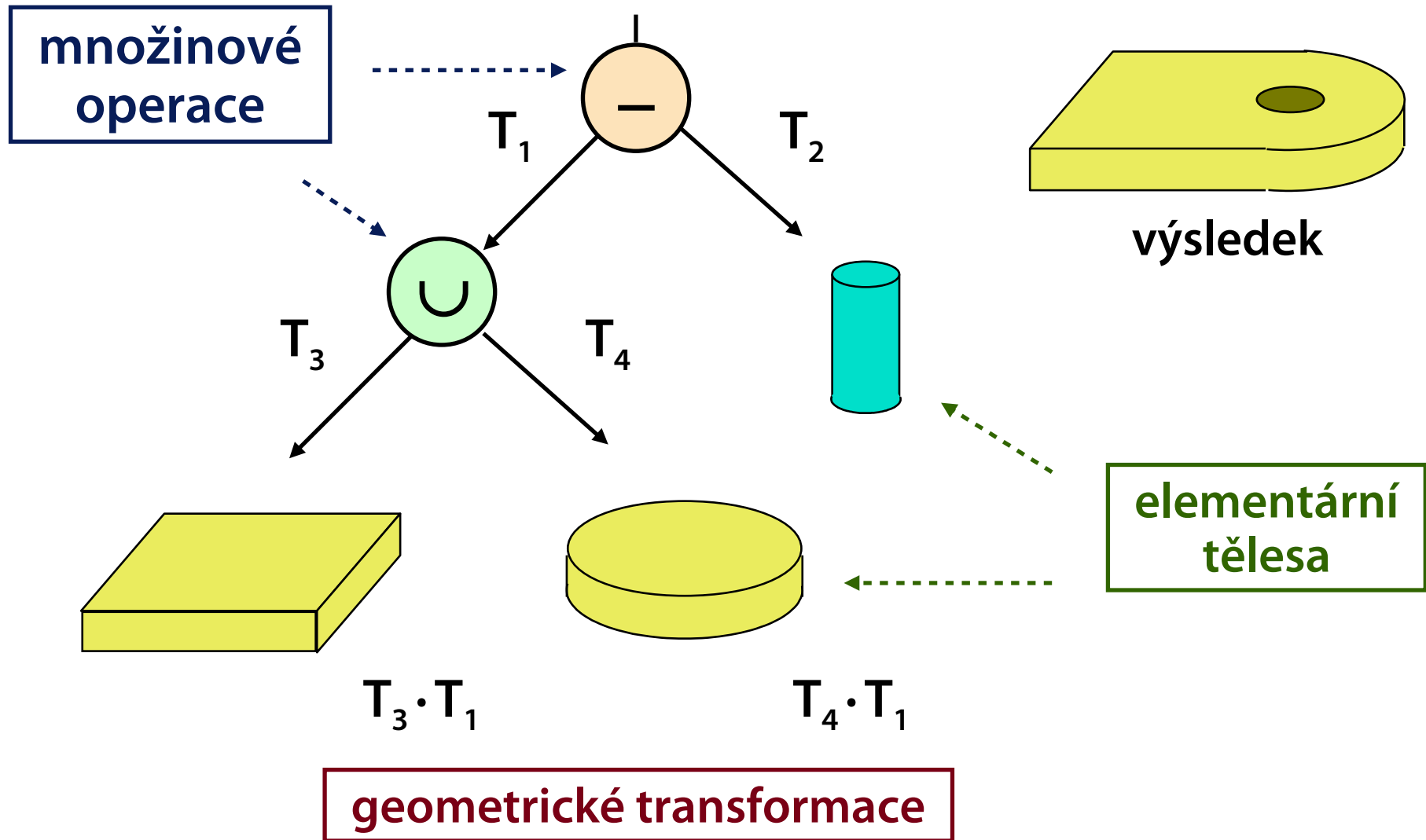
- kompozice složitějších těles z elementárních
- sjednocení, průnik, rozdíl...

Geometrické transformace

- modifikace elementárních i složitějších těles
- homogenní maticové transformace



CSG strom





Transformace v CSG stromu

Význam (sémantika) transformace T_i

- T_i mohou být uloženy v každé hraně CSG stromu
- převod souřadnic ze soustavy podtělesa (podstromu, elementárního tělesa) do soustavy nadtělesa
- „podtěleso transformuji pomocí T_i před tím, než ho přidám do nadtělesa“

Snadná transformace libovolného podstromu

- změním pouze jednu matici

Inverzní transformace T_i^{-1}

- pro vyčíslovací algoritmy (test „bod×CSG“, zobrazení)



Transformace v CSG stromu

Uložení transformací jen v listech

- kumulované součiny (např. $T_3 \cdot T_2 \cdot T_1$ nebo inverzní $T_1^{-1} \cdot T_2^{-1} \cdot T_3^{-1}$)
- urychlení vyčíslovacích algoritmů (pro editaci je výhodnější distribuované uložení transformací)

Úsporné uložení elementárních těles

- tělesa jsou uložena v **normovaném tvaru**, všechny změny se provádí geometrickými transformacemi
 - » krychle (jednotková, vrchol v počátku)
 - » koule (jednotková, střed v počátku)
 - » válec (vodorovná podstava – jednotkový kruh, svislá osa, výška 1)
 - » ...



Test „bod \times CSG strom“

Leží daný bod A uvnitř tělesa?

- někdy chceme zjistit i podtělesa obsahující bod A

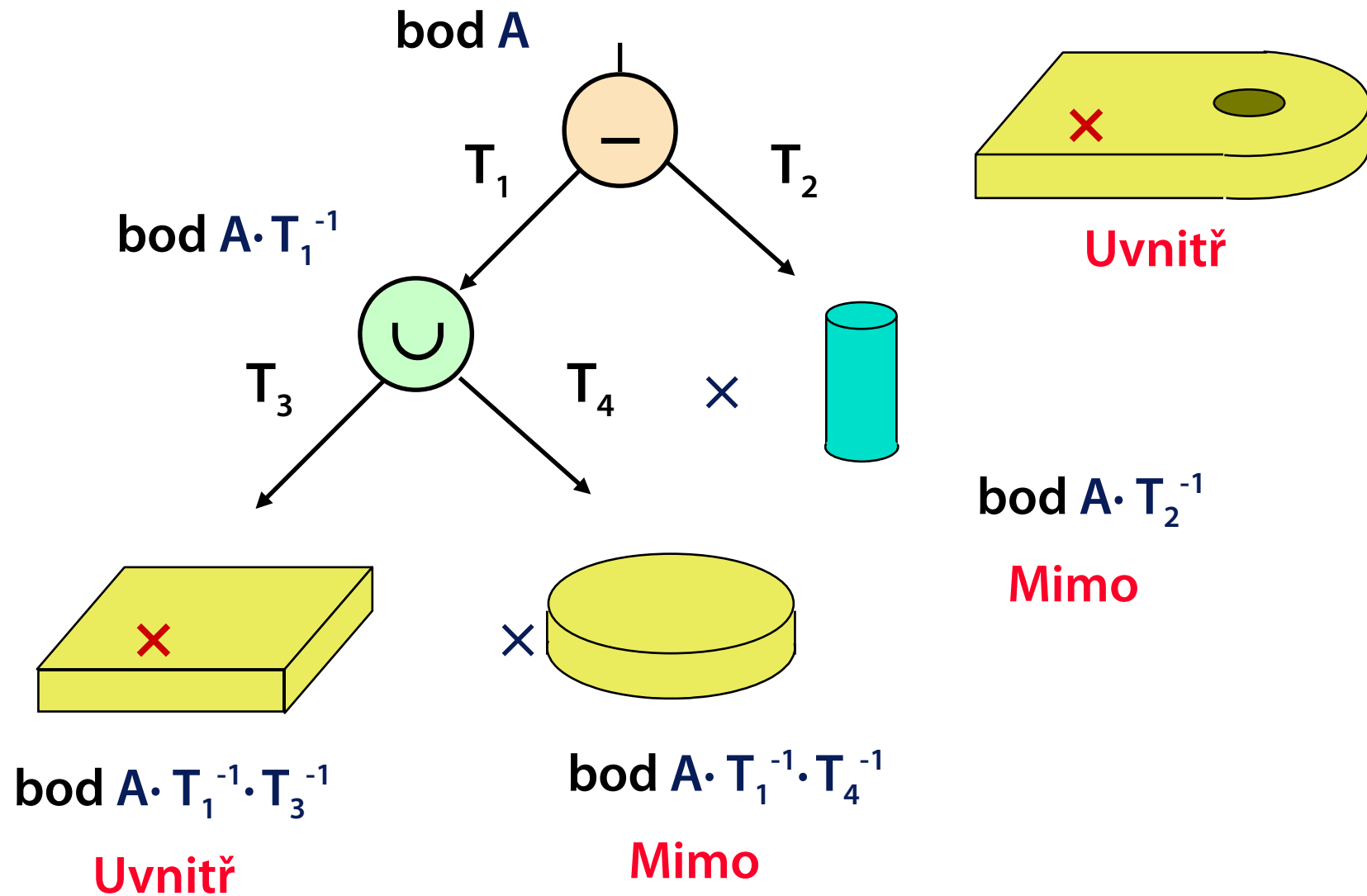
Testy „bod \times elementární těleso“ jsou snadné!
(především pro normované tvary těles)

Průchod CSG stromem

- souřadnice bodu A se převádějí do souřadných soustav elementárních těles (inverzní transformace)
- místo množinových operací se provádějí jejich **boolovské ekvivalenty** (\vee místo \cup , \wedge místo \cap ...)



Test „bod \times CSG strom“





Zobrazování CSG reprezentace

Převedení do povrchové reprezentace

- pro každý druh **elementárního tělesa**: rutina převádějící těleso na **mnohostěn**
- **množinové operace nad mnohostěny** (omezená přesnost – výsledek nemusí být správně ani v topologickém smyslu)

Vrhání paprsku („Ray-casting“)

- přesné zobrazování v **rastrovém prostředí** (pixelová přesnost)
- výpočetně náročnější metoda



Povrchové reprezentace

„Drátový model“

- pseudo-povrchová reprezentace
- pouze **vrcholy** a **hrany** těles (nelze použít pro výpočet viditelnosti)

VHS(T) reprezentace

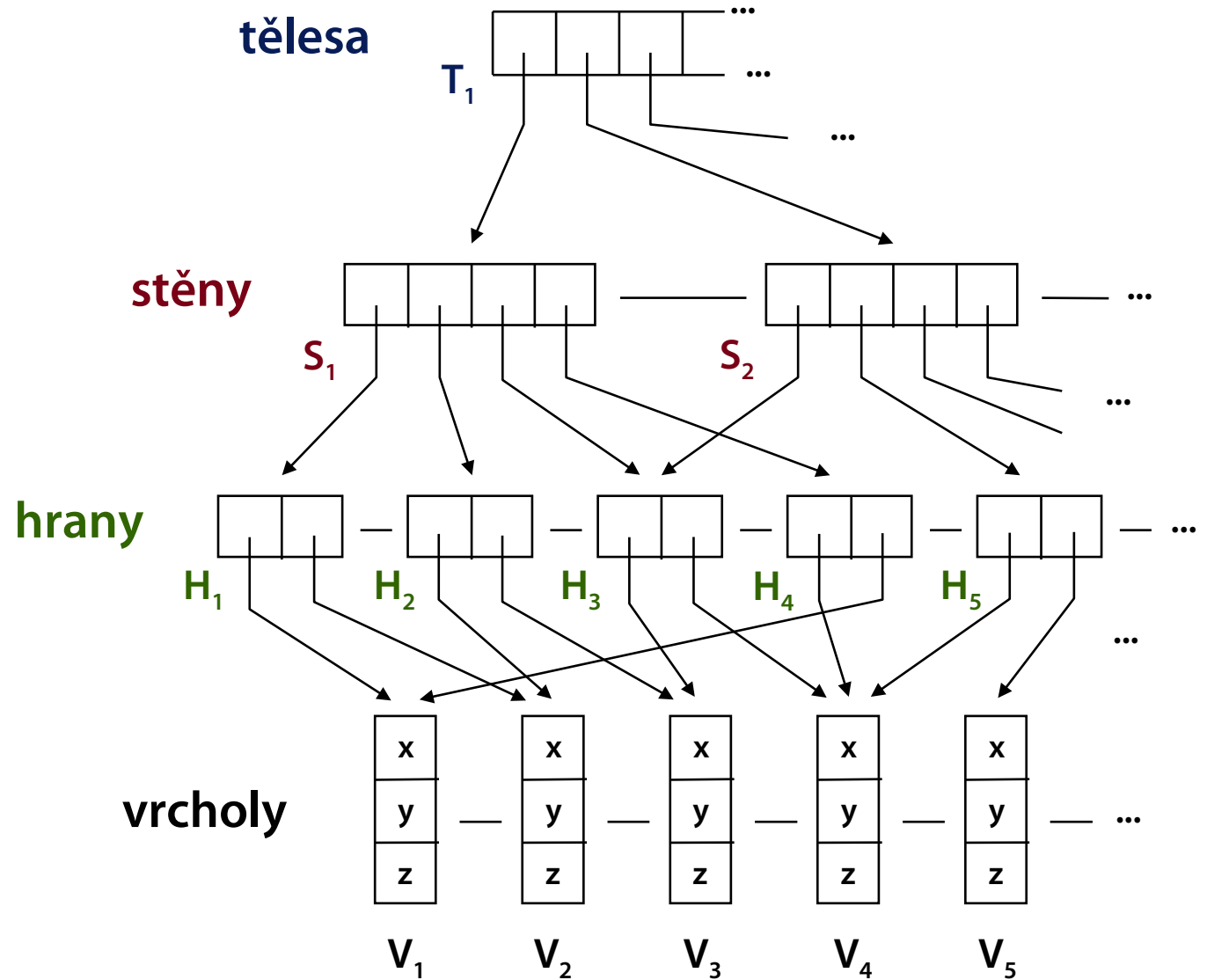
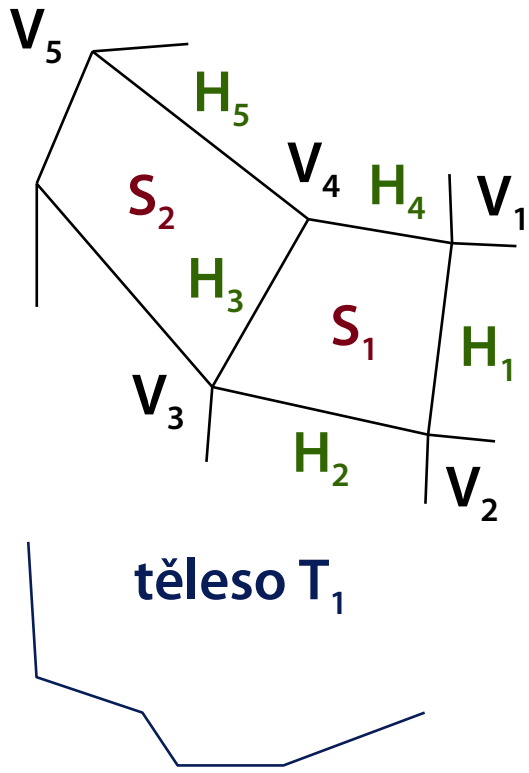
- kompletní topologická informace: seznamy **vrcholů**, **hran**, **stěn** (a **těles**)

„Okřídlená hrana“ („winged-edge“)

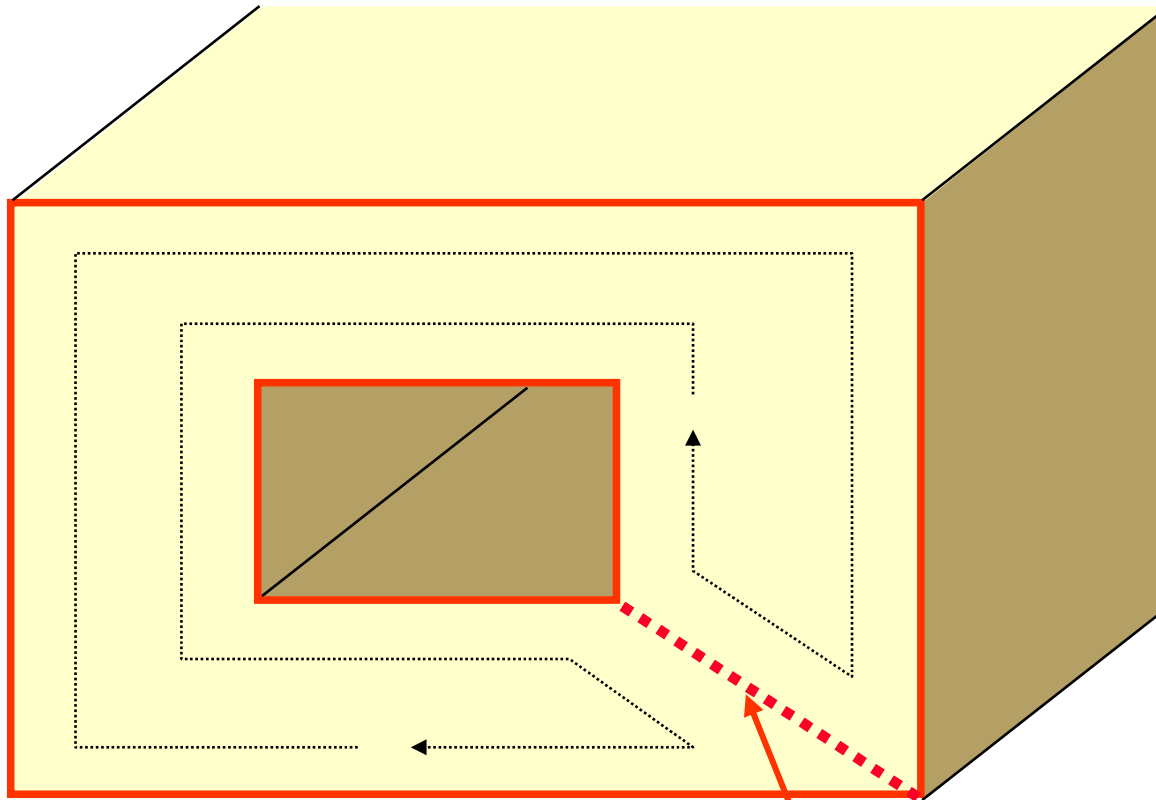
- redundantní informace pro **rychlé vyhledávání** sousedních objektů (hrany incidentní s vrcholem...)



Povrchová reprezentace VHST



„Děravá“ stěna

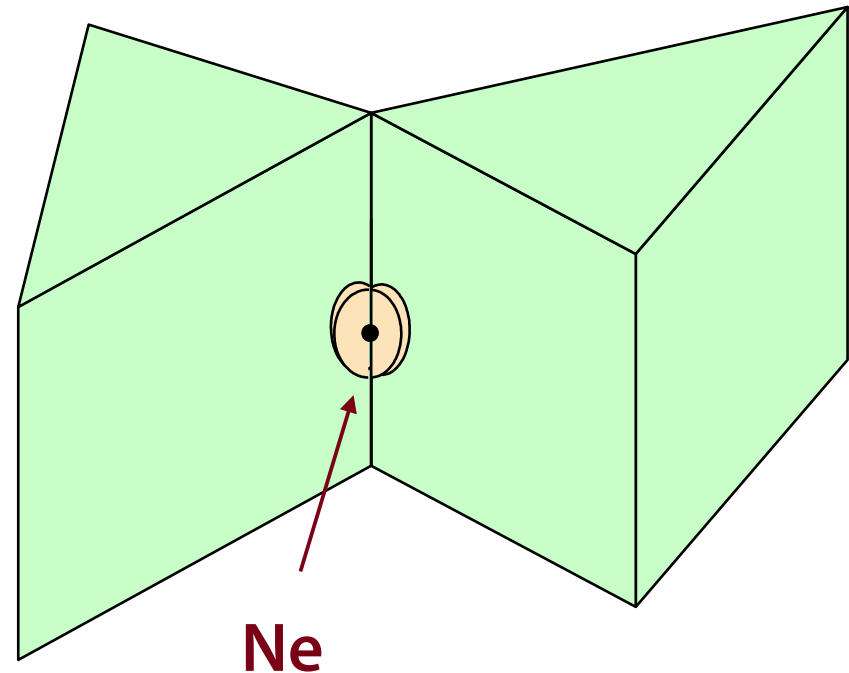
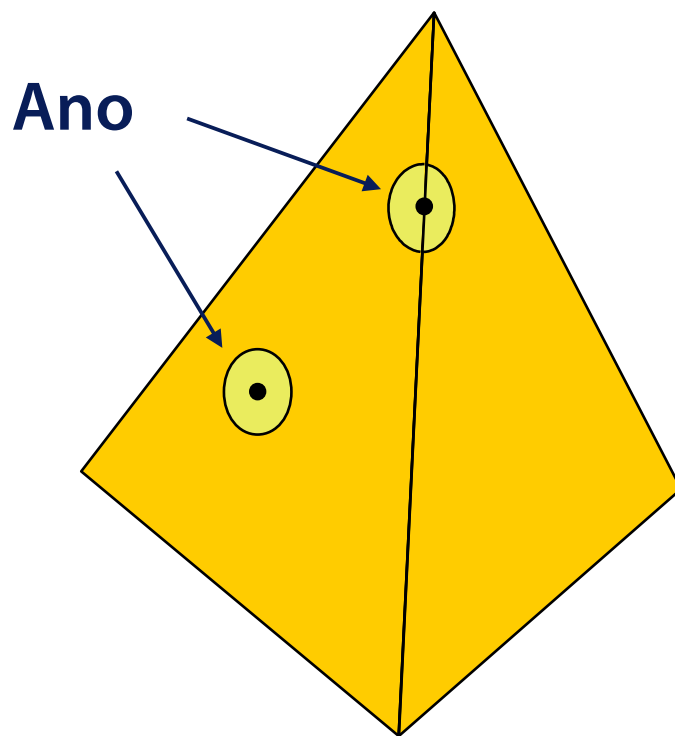


**umělá hrana
(nevykresluje se)**



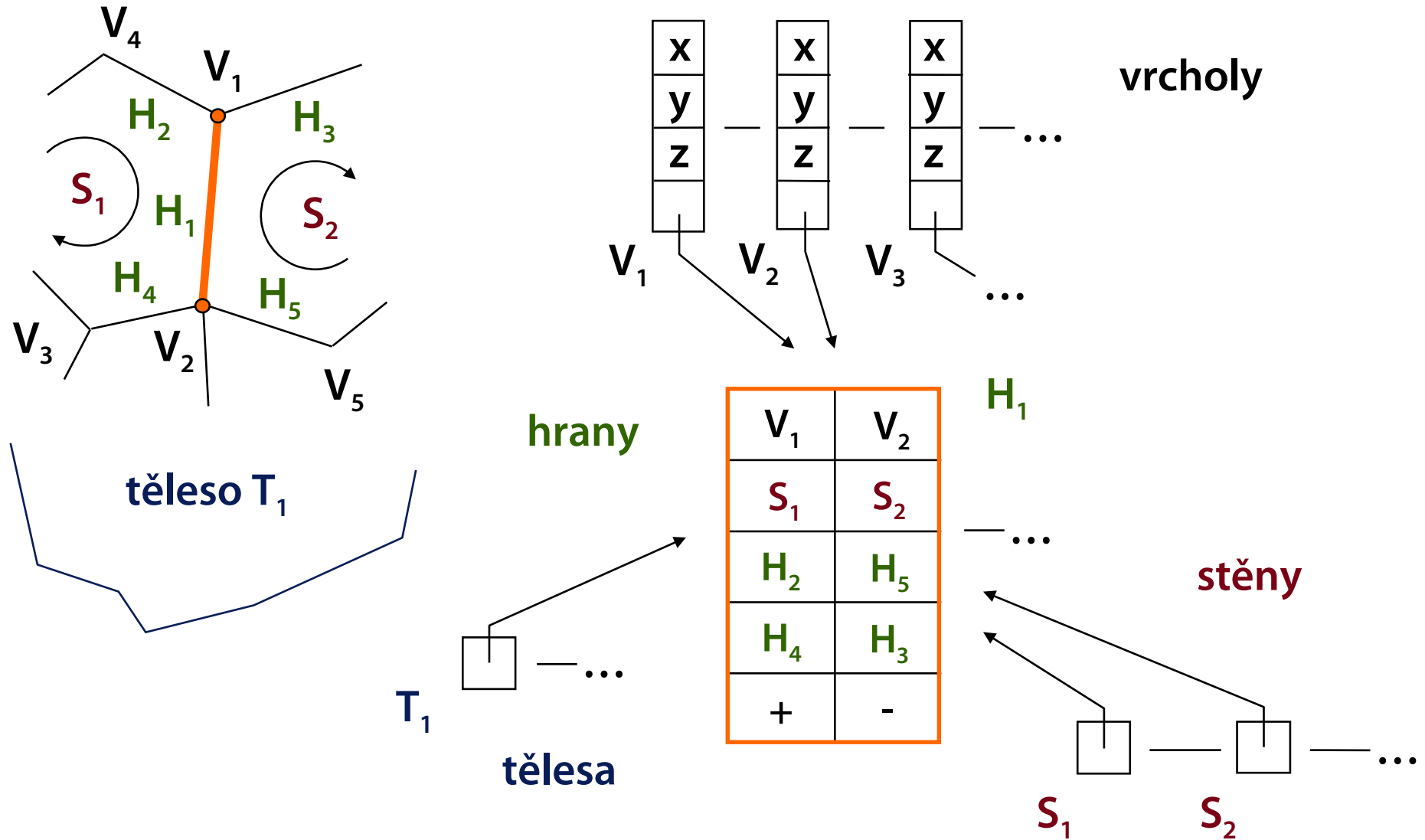
„2-manifold“ (manifold, varieta dim=2)

Def: Pro každý povrchový bod existuje okolí, které je topologicky ekvivalentní s rovinou





Okřídlená hrana („winged-edge“)





Další informace v databázi

Vrchol (vertex, V)

- barva, texturové souřadnice
- normálový vektor (stínování hladké plochy)

Hrana (edge, E)

- příznak umělé hrany (pro reprezentaci děravých stěn)

Stěna (face, F)

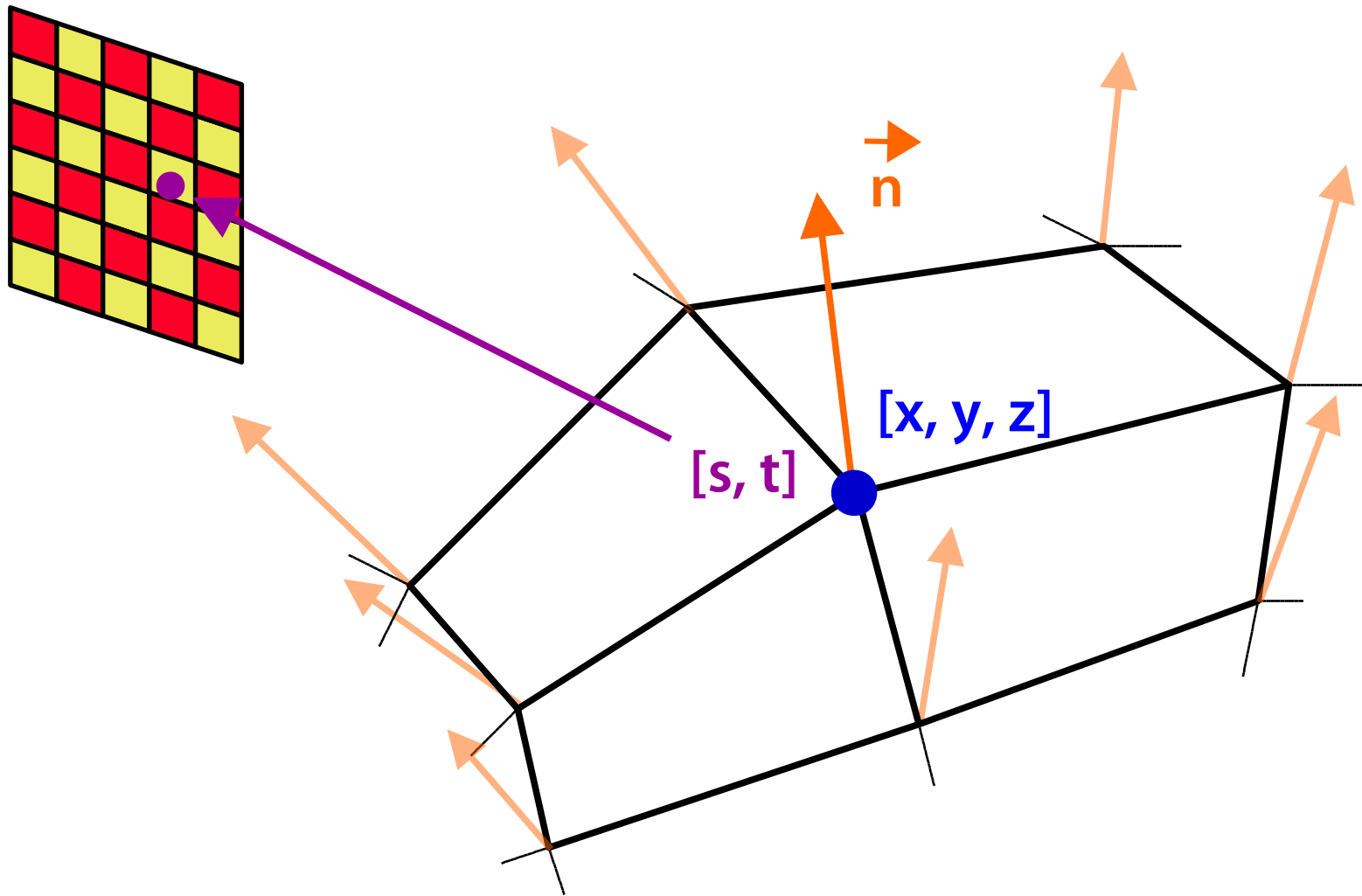
- barva, materiál, textura
- normálový vektor (stínování, přivrácená/odvrácená)

Těleso (solid, S)

- barva, materiál, textura



Data ve vrcholu





Data ve vrcholu

Souřadnice

- $[x, y, z]$ nebo $[x, y, z, w]$

Normálový vektor

- pro stínování (později), může být interpolován

Barva

- explicitní barva povrchu, může být interpolována

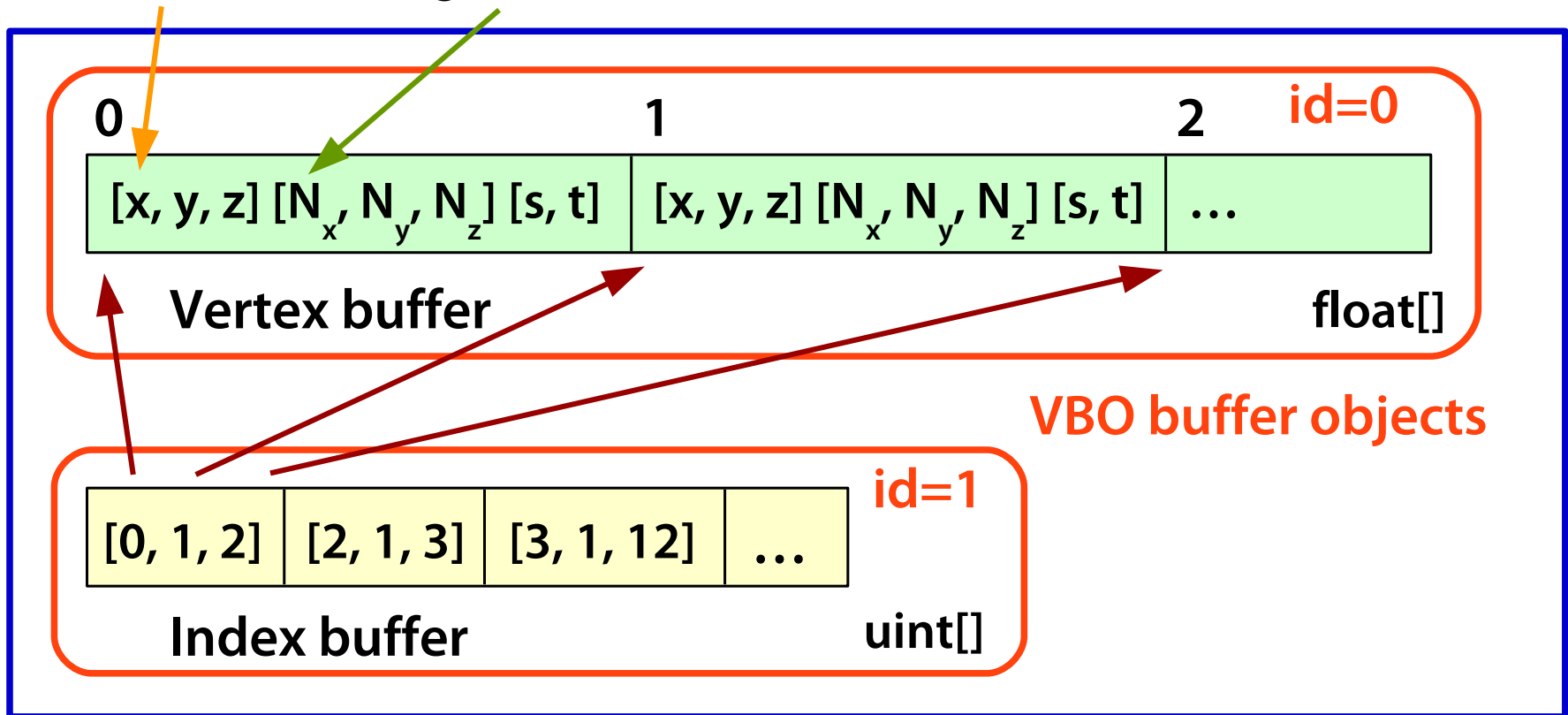
Texturové souřadnice

- nejčastěji 2D souřadnice – $[s, t]$ nebo $[u, v]$
 - » normalizovaný prostor textury $[0, 1]^2$
- automaticky se při kreslení interpolují do pixelů



Vertex Buffer Objects (Buffers)

```
glBindBuffer(GL_ARRAY_BUFFER, 0);  
glVertexPointer(...); glNormalPointer(...); ...
```



```
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 1);  
glDrawElements(GL_TRIANGLES, ...);
```

GPU memory



„Corner Table“ (trojúhelníky)

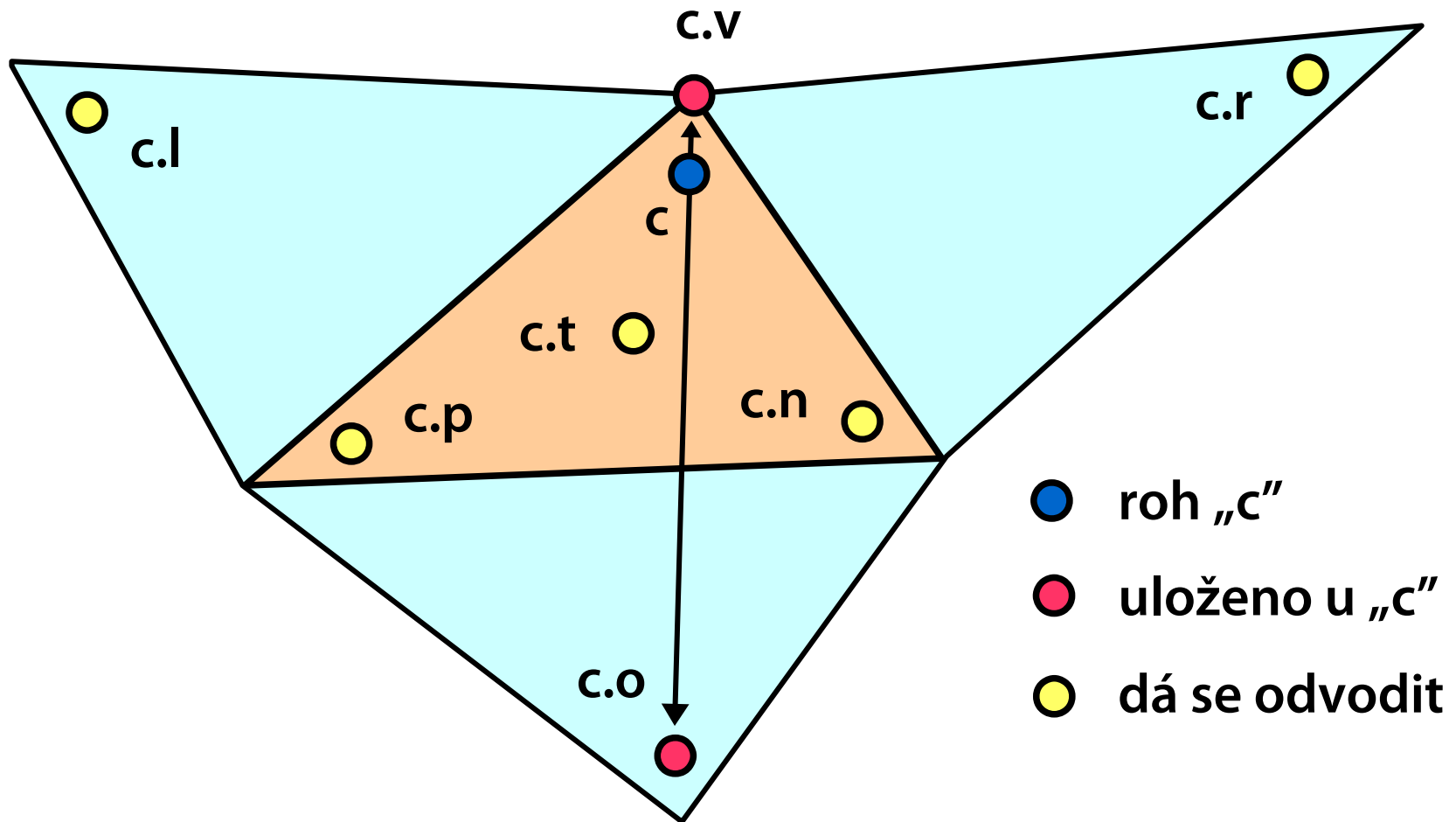
Tabulka vrcholů $G[v]$

- souřadnice, normála, barva, texturové souřadnice...

Tabulka rohů $V[c]$

- jeden vnitřní roh trojúhelníka
- index vrcholu („**c.v**“)
- rohy jsou uloženy za sebou v 1D poli, CW orientace stěn
- protější roh protějšího trojúhelníka („**c.o**“)
- implicitní údaje, navigace
 - » číslo trojúhelníka $t = c \text{ div } 3$
 - » ostatní rohy $c.n = (c \bmod 3 == 2) ? c-2 : c+1, c.p = c.n.n$
 - » další sousední trojúhelníky $c.l = c.n.o, c.r = c.p.o$

„Corner Table“





„Corner Table“ – kódování

Tabulka „o“ se nemusí přenášet

- lze ji jednoznačně rekonstruovat z „v“

Hodnoty indexů „v“ se mohou bitově ořezat

- index vrcholu obsahuje $31 - \log_2 v$ úvodních nul

Ořezání souřadnic

- obalový kvádr celého objektu, uvnitř se používají relativní souřadnice (10 až 16 bitů na složku)
- predikce polohy vrcholů – při inkrementálním průchodu daty (např. „Edgebreaker“) – další úspory



„Edgebreaker“ (Rossignac 1999)

Úsporné kódování tri-mesh pomocí inkrementálního průchodu sítí

- **pozice vrcholů** se komprimují za pomoci predikce (až 7 bpv)
- **topologie sítě** se ukládá velice úsporně na základě průchodu (1.0 až 1.8 bpv)

„CLERS“ kód

- **pět možností**, jak pokračovat z aktuálního trojúhelníka
- velký potenciál entropické komprese (některé kroky/posloupnosti jsou mnohem pravděpodobnější)



Eulerovy zákony

Pro jednoduchý polyedr (bez děr) platí vzorec

$$V - H + S = 2$$

V – počet vrcholů, H – počet hran, S – počet stěn

Zobecněný vzorec (dovoluje díry)

$$V - H + S - D = 2 (T - G)$$

D – počet děr ve stěnách, T – počet těles, G – počet děr procházejících celým tělesem (Genus)



Eulerovy operátory

Konstrukce 2-manifoldsu po krocích

- v každém kroku je zajištěna platnost Eulerových vzorců (těleso je topologicky korektní)
- ke každému operátoru existuje inverzní (snadná implementace příkazu „Undo“)

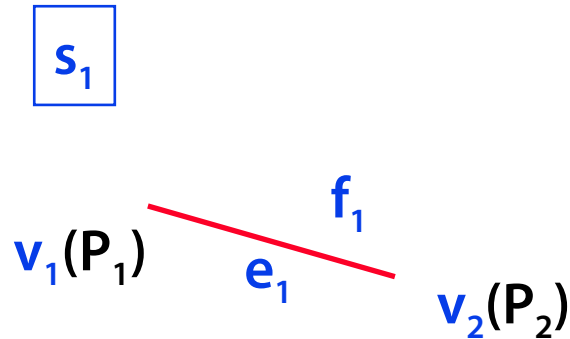
Příklady Eulerových operátorů

- **Msfevv**(P_1, P_2) „make solid, face, edge, vertex, vertex“
- **Mev**(f_1, v_1, P_2) „make edge, vertex“
- **Mef**(f_1, v_1, v_2) „make edge, face“
- **Kef**(e) „kill edge, face“

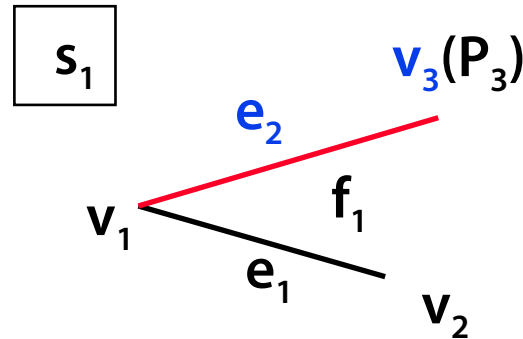


Konstrukce čtyřstěnu

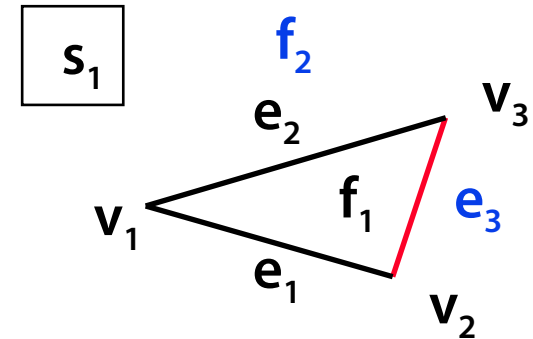
1. $M_{sf}v_v(P_1, P_2)$



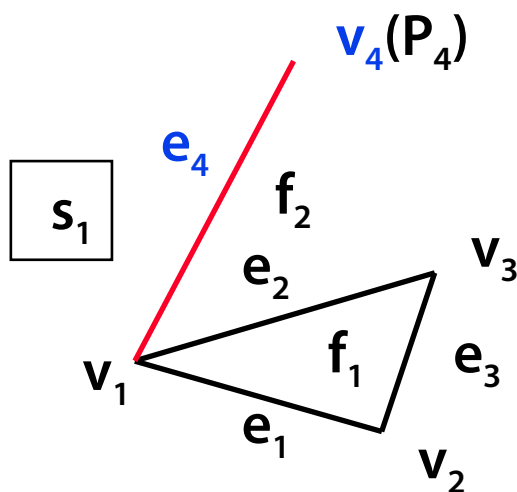
2. $M_{ev}(f_1, v_1, P_3)$



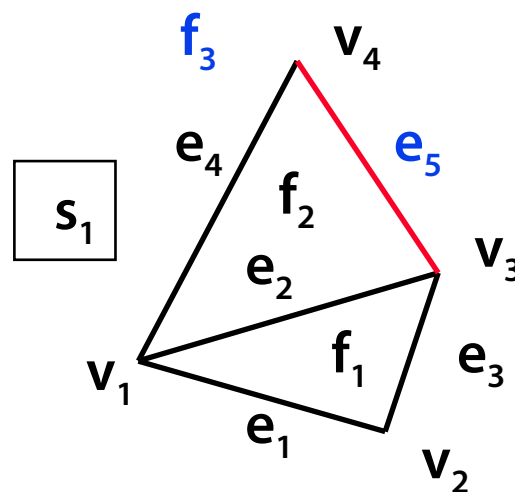
3. $M_{ef}(f_1, v_2, v_3)$



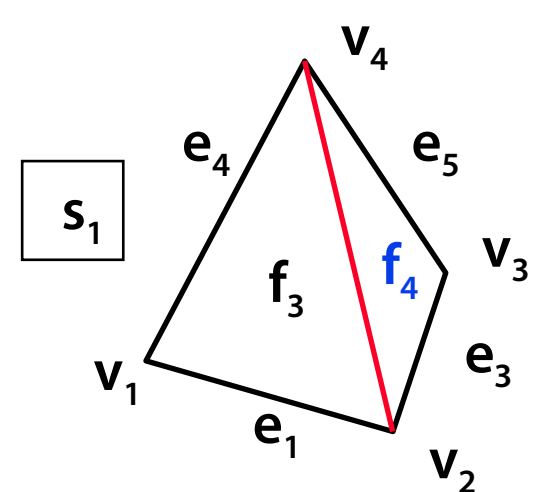
4. $M_{ev}(f_2, v_1, P_4)$



5. $M_{ef}(f_2, v_3, v_4)$

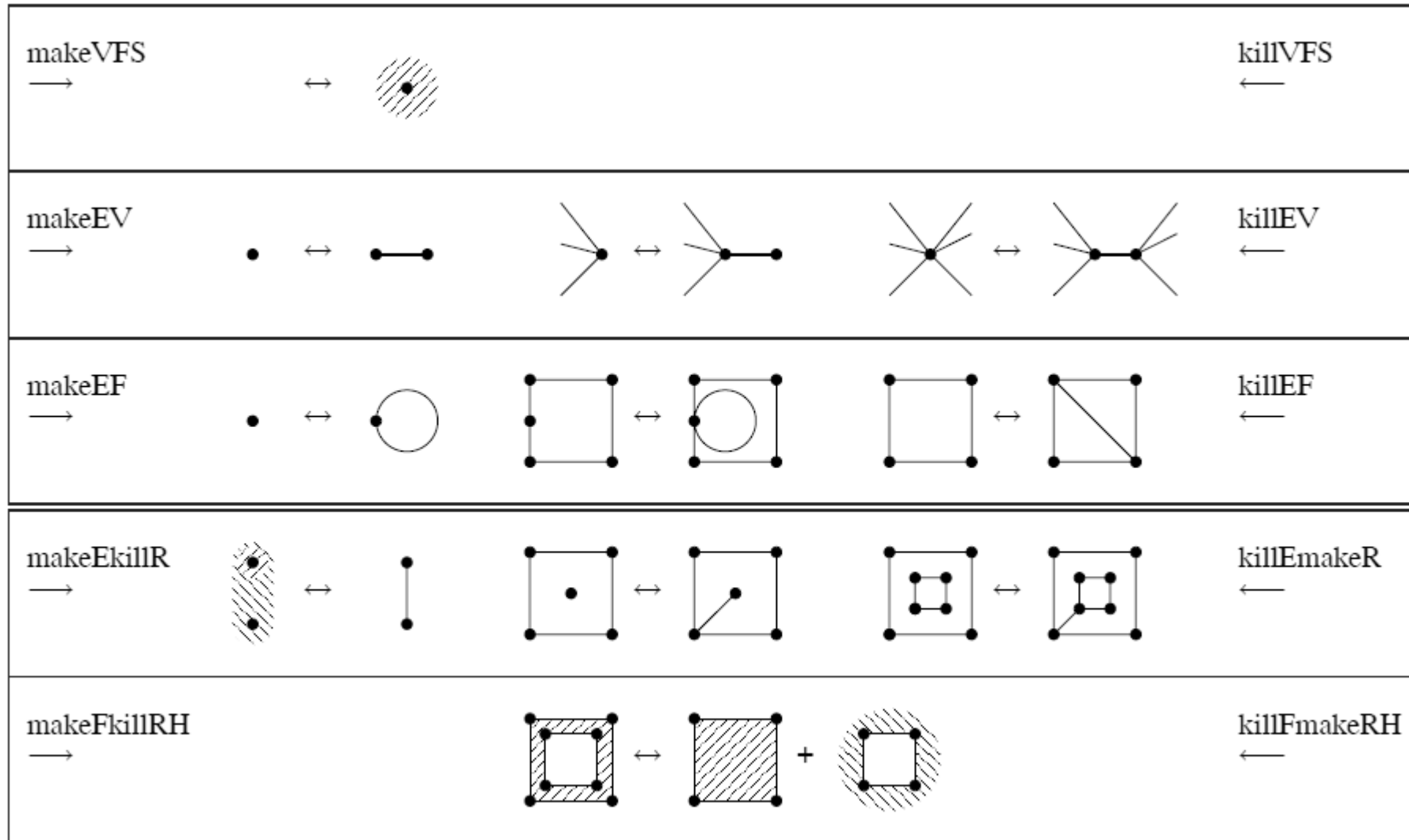


6. $M_{ef}(f_3, v_2, v_4)$





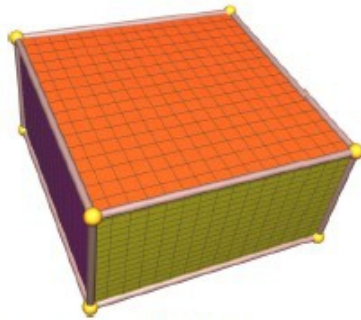
Rozšířená sada operací



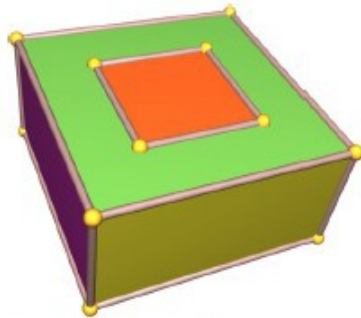
Krychle s otvorem



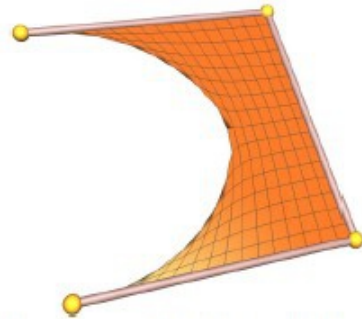
1: makeVFS



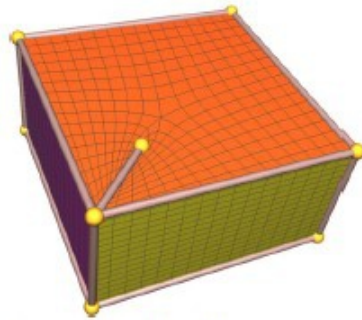
5: 4 × makeEF (c)



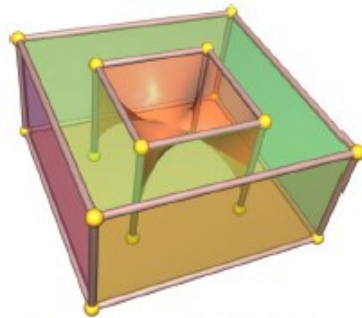
9: killEmakeR (c)



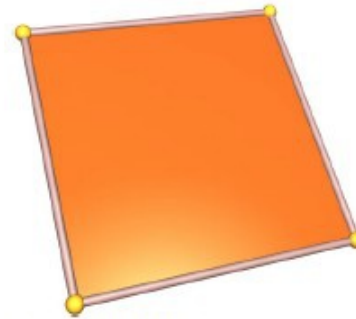
2: 3 × makeEV (a,b,b)



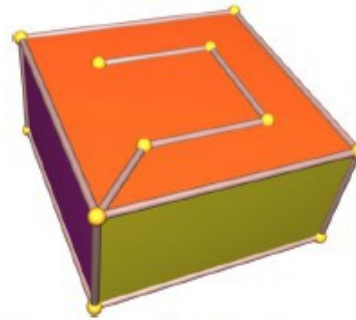
6: makeEV (b)



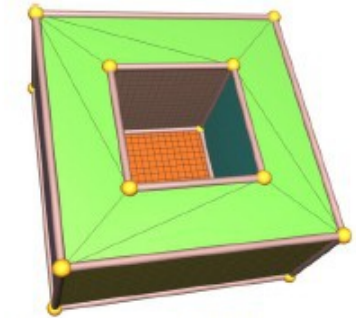
10: 4 × makeEV (b)



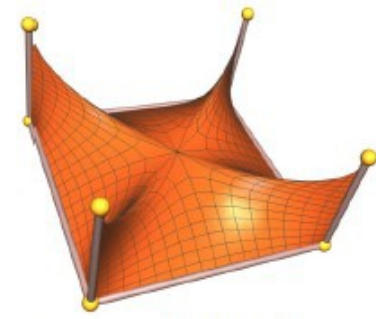
3: makeEF (c)



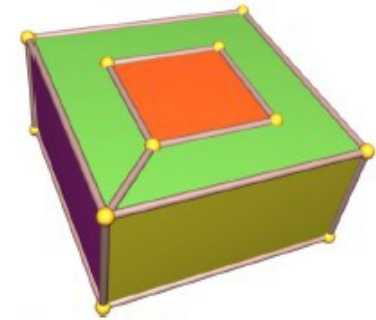
7: 3 × makeEV (b)



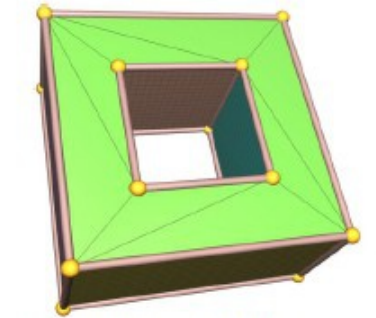
11: 4 × makeEF (c)



4: 4 × makeEV (b)



8: makeEF (c)



12: killFmakeRH



Literatura

J. Foley, A. van Dam, S. Feiner, J. Hughes: *Computer Graphics, Principles and Practice*, 534-562, 712-714

Jiří Žára a kol.: *Počítačová grafika, principy a algoritmy*, 234-238

J. Rossignac, A. Safanova, A. Szymczak: *3D compression made simple: Edgebreaker on a Corner Table*, 2001

H. Lopes, J. Rossignac, A. Safonova, A. Szymczak and G. Tavares: *Edgebreaker: A Simple Compression Algorithm for Surfaces with Handles*, C&G Intl. J, vol.27(4), 553-567, 2003

Sven Havemann: *Generative Mesh Modeling*, PhD thesis, 2005, TU Braunschweig, pp. 59-79