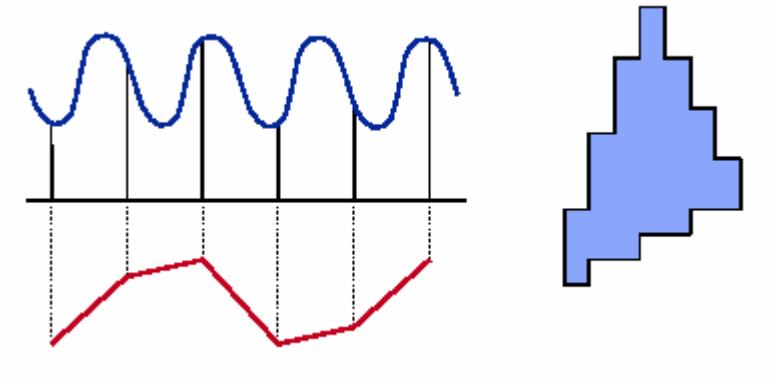
Rekonstrukce a vzorkování

1. Vysvětlete pojem (jev) „alias“. V jakých situacích vzniká, co je jeho příčinou a důsledkem? Uveďte metody používané pro antialising včetně vysvětlení základního principu?

„alias“ – rušivý jev vzniklý zobrazením v pravidelné diskrétní mřížce



Prostorový alias
- **zubaté** zobrazení šikmých linií
  - při kreslení husté sítě čar vzniká tzv. „Moiré efekt“
- **interference** rechle se měnícího obrazu s pixelovým rasterem
  - příklad: plot v perspektivní projekci
  - příliš jemná nebo příliš vzdálená pravidelná textura (šachovnice ve velké vzdálenosti)

Časový alias
- projevuje se zejména při animaci **pomalého pohybu**
- **blikání** na obvodu pohybujících se objektů
  - v extrémním případě se celé malé objekty objevují a opět mizí
- **interference** cyklického pohybu se snímkovou frekvencí
  - otáčející se kolo se zdánlivě zastaví nebo se pomalu točí opačným směrem

Vzorkování a rekonstrukce
- **vzorkování** nebo **výpočet** obrazové funkce
  - před vzorkováním by se z obrazu měly odstranit všechny vyšší (nezobrazitelné) frekvence
  - filtr typu dolní propustnost(průměrování v okénku)
  - při **syntéze obrazu** se mohou vyšší frekvence zanedbat přímo (vyhlazování vzorkováním plochy)
- **rekonstrukční filtr** je dán vlastnostmi výstupního zařízení
  - např. na monitoru se stopy sousedních pixelů překrývají

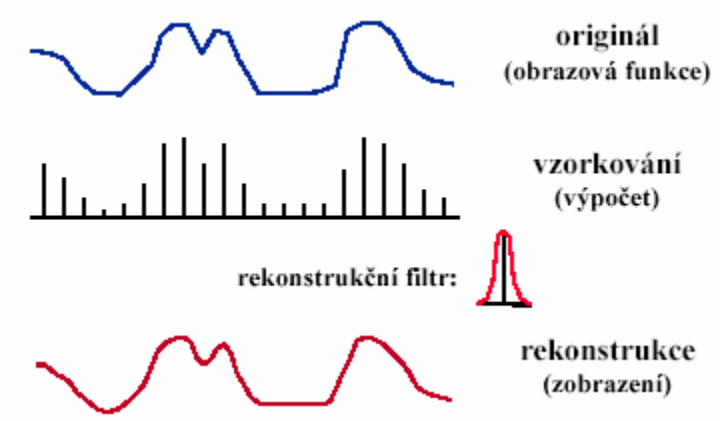
Já: Metody pro odstranění aliasu
- odstranění(odfiltrování) vysokých frekvencí
  - většinou vzorkování s vyšší frekvencí (supersampling), následná filtrace (rozmazání) a převedení zpět na původní vzorkování
  - nebo odstranění vysokých frekvencí přímo ze spojitého obrazu
- převedení aliasu na šum(stochastické vzorkování)
  - vzorkování s vyšší frekvencí ale ne pravidělně (Poisonovo rozložení, roztřesení, částečné roztřesení, náhodné vzorkování, nezávislé roztřesení)

2. Co je rekonstrukční filtr? Jak ovlivňuje zobrazení vzorkovaného signálu ?

- **rekonstrukční filtr** je dán vlastnostmi výstupního zařízení
  - o např. na monitoru se stopy sousedních pixelů překrývají

Podmínky rekonstrukce

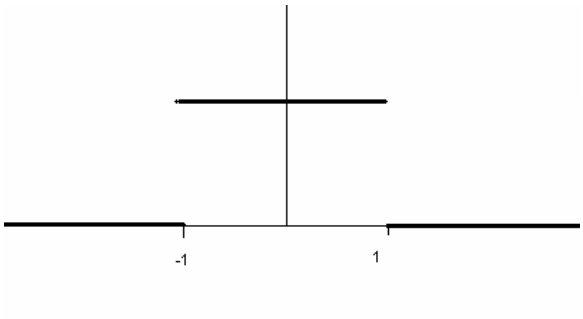
Signál musí být frekvenčně omezen vzorkovací frekvence f_s musí být větší než dvojnásobek nejvyšší frekvence f_{max} přítomné v signálu (Nyquistova rychlost)



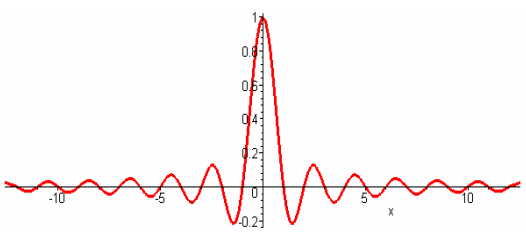
3. Co je spektrum neperiodické funkce ? Jak ovlivňuje pravidelné vzorkování spektrum výsledné, tj. vzorkované funkce ? Jaké jsou podmínky pro úplnou rekonstrukci vzorkovaného signálu ?

4. Jaký tvar má ideální nízkopásmový filtr ? K čemu se používá ? Jaký konvoluční filtr je jeho obrazem ? Nakreslete a vysvětlete na příkladu vzorkovaného signálu s omezeným frekvenčním rozsahem.

MPG:
Ideální filtr vysokých frekvencí má ve Fourierově prostoru tvar skokove funkce se středem v počátku (-1..1->1, jinak 0). Jemu v obrazové doméně odpovídá funkce sinc(x). Protože je funkce sinc(x) neomezená, musí se v praxi aproximovat.



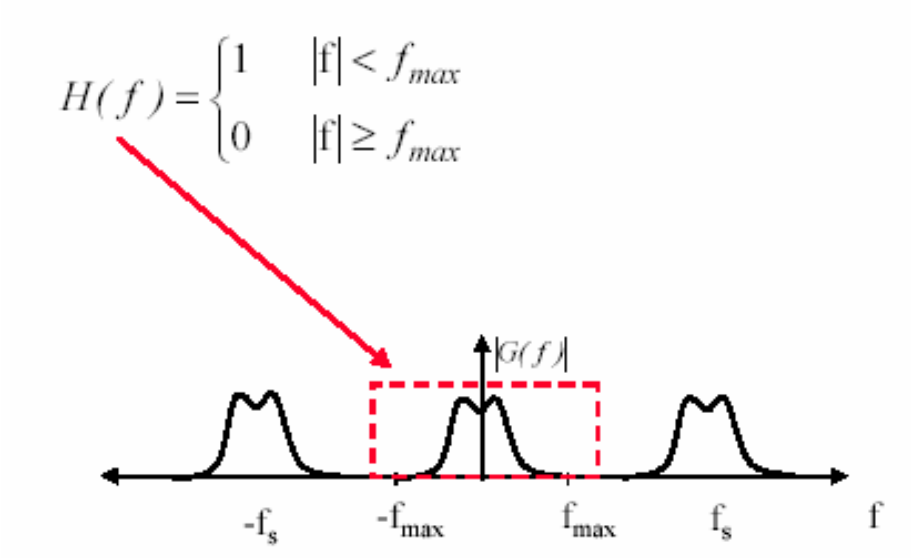
skoková funkce(Fourieruv prostor)



sinc(x) (obrazová doména)

Ideální nízkopásmový filtr

$$H(f) = \begin{cases} 1 & |f| < f_{max} \\ 0 & |f| \geq f_{max} \end{cases}$$



Ideální rekonstrukce

$$H(f) \leftrightarrow sinc(x) = \frac{sin(\pi x)}{\pi x}$$

$$A/2W, 2W \leftrightarrow A\frac{sin(\pi x / W)}{\pi x / W}$$

Rekonstrukce (podle konvol.teorému):

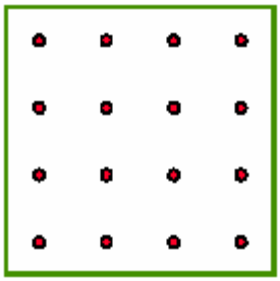
$$g(x) = \sin c(x) * g_s(x)$$

$$= \int_{-\infty}^{\infty} \sin c(\lambda) g_s(x - \lambda) d\lambda$$

5. Jaké vlastnosti by měl mít dobrý vzorkovací algoritmus ? Uveďte příklady vzorkovacích algoritmů, jejich principy a vlastnosti.

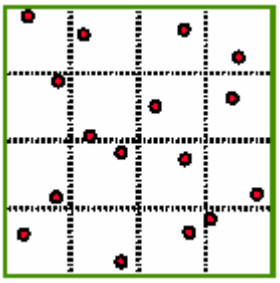
Vlastnosti vzorkovacího algoritmu:
- rovnoměrné pokrytí dané oblasti
- absolutní pravidelnost je nežádoucí (interference)
- efektivní výpočet

Vzorkovací algoritmy:
- Pravidelné vzorkování
    - Neodstraňuje rušivé interference (pouze je přesune do vyšších frekvencí)

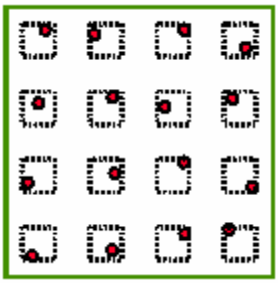


- Náhodné vzorkování
  - N nezávislých náhodných pokusů s rovnoměrným rozložením pravděpodobnosti
  - Vzorky mohou vytvářet **větší shluky**
  - Velký podíl **šumu** ve výsledku

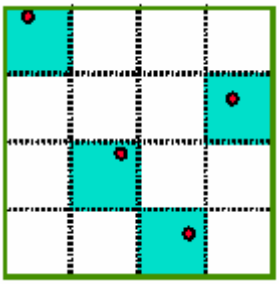
- „roztřesení“ („jittering“)
  - K x K nezávislých náhodných pokusů v K x K shodných subintervalech (pokrývajících původní interval beze zbytku)
  - Omezení pravděpodobnosti **velkých shluků**
  - **Rovnoměrnější pokrytí** vzorkovaného intervalu



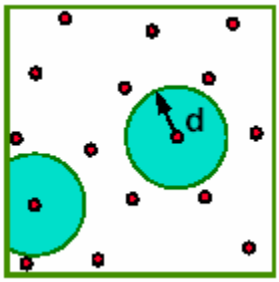
- Částečné „roztřesení“ („semijittering“)



- N věží (nezávislé „roztřesení“)



- Poissonovo diskové vzorkování



6. Praktické vyhlazovací metody.

Metody vyhlazování
1. úprava signálu – **prefiltering**
   redukce šířky pásma signálu pomocí nízkofrekvenčního filtru
   nejkvalitnější, ale často nepraktické řešení
2. úprava vzorků pomocí **supersampling**
   pomocí více vzorků zvýší Nyquistovu frekvenci
   jednoduchá, často používaná metoda
3. úprava vzorků pomocí **stochastického vzorkování**
   relativně jednoduchá metoda, používá se v kombinaci s 2.

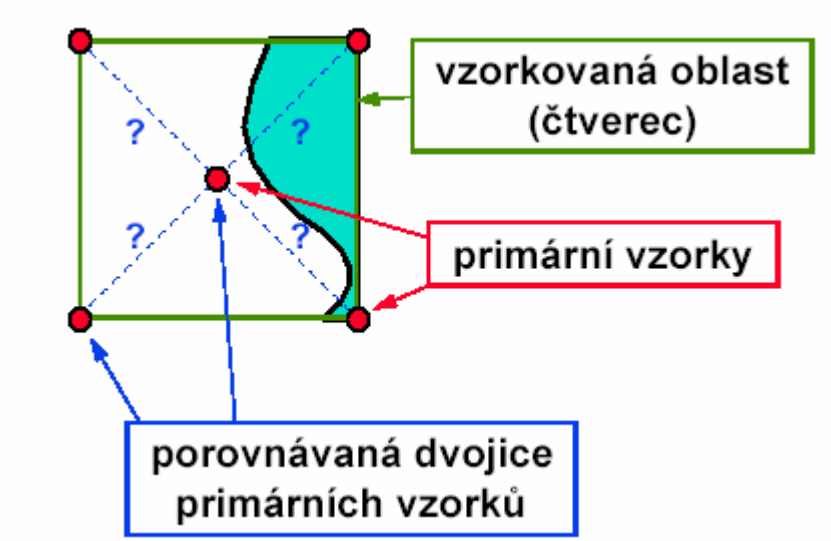
Vzorkovací metody
- **předpis: k … $[x_k, y_k]$**
  - výběr vzorku z dané oblasti:
    nejčastěji tvaru obdélníka, čtverce nebo kruhu v 2D
  - vzorkování ve vyšších dimenzích (řádově do dim=10)
- požadované **vlastnosti** vzorkovacího algoritmu
  - rovnoměrné pokrytí dané oblasti
  - absolutní pravidelnost je nežádoucí (interference)
  - efektivní výpočet

7. Vzorkování s adaptivním zjemňováním. Kritéria, rekonstrukce výsledku.

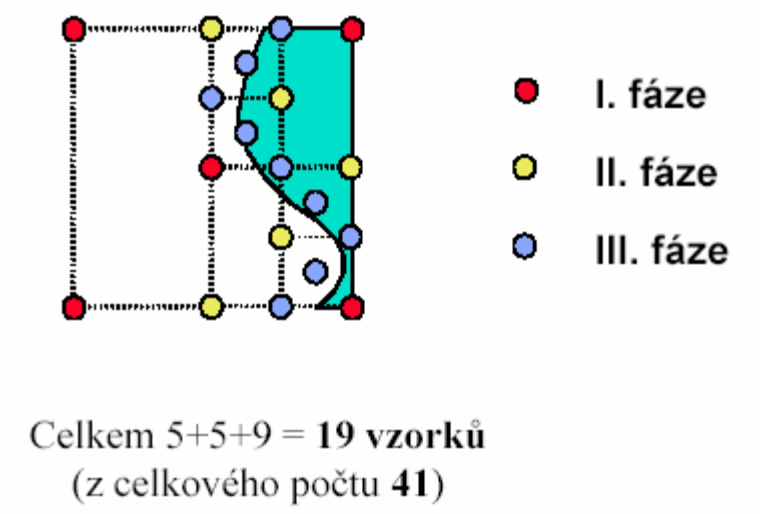
Adaptivní zjemňování
- vzorkování podle **lokální důležitosti** (vážené vzorkování) nebo **zajímavosti**
  - některé oblasti pokrývaného intervalu mají větší váhu
  - oblasti s větší variací vzorkované funkce je nutno pokrýt hustěji
- „důležitost“ nebo „zajímavost“ **nemusím znát dopředu**
  - algoritmus se musí přizpůsobovat dosaženým výsledkům (adaptabilita)

Zjemňovací kritéria
- **funkční hodnoty** (rozdíl, rozptyl, gradient)
  - rozdíl v barvě sousedních vzorků, …
- **čísla zobrazených těles**
  - větší priorita
  - textury s opakujícími se vzory: signatury
- **stromy výpočtu** (rekurzivní sledování paprsku)
  - topologické porovnávání celých stromů nebo jen několika horních pater
  - identifikátor stromu – rekurzivní konstrukce pomocí našívací funkce

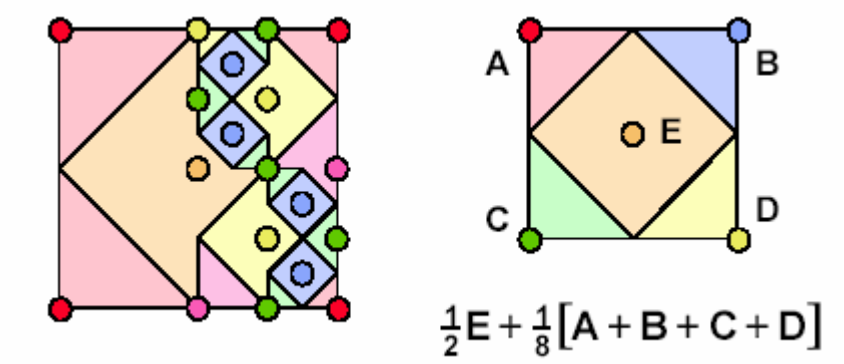
Rekurzivní zjemňování (Whitted)



Výsledný soubor vzorků



Rekonstrukce výsledku



V každém již dále neděleném čtverci se plocha rozdělí na dva protější vzorky

Textury

8. Uveďte jednotlivé typy používaných textur. Co ovlivňuje textura v lokálním světelném modelu a lokální geometrii tělesa ? Zapište jednoduchý světelný model a naznačte možnosti aplikace textury.

Já: Textury

- 2D textury
- 3D textury
- Bump textury (hrbolaté textury)

Co mění textura

- Nejčastější:
  - Difúzní barva

$$I_{diff} = I_a \rho_a + I_{light}(\rho_d \cos\theta + \rho_s \cos^n \alpha)$$

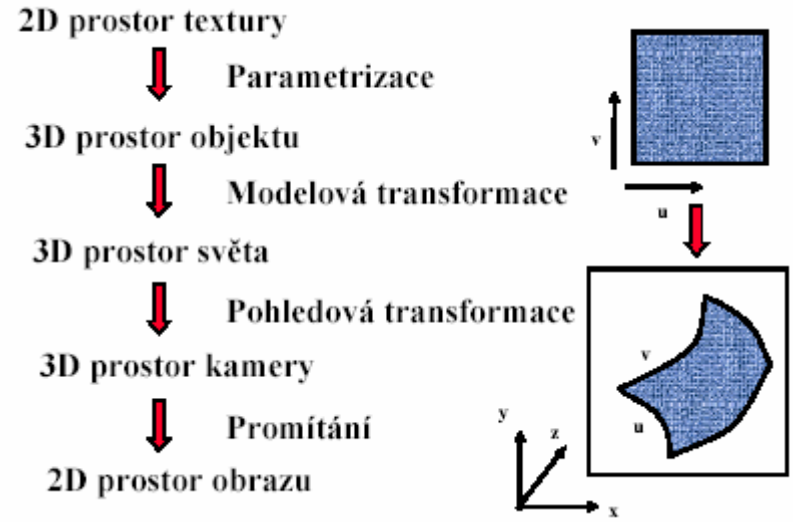
  - A tak … zrcadlové efekty se nemění

Ale …

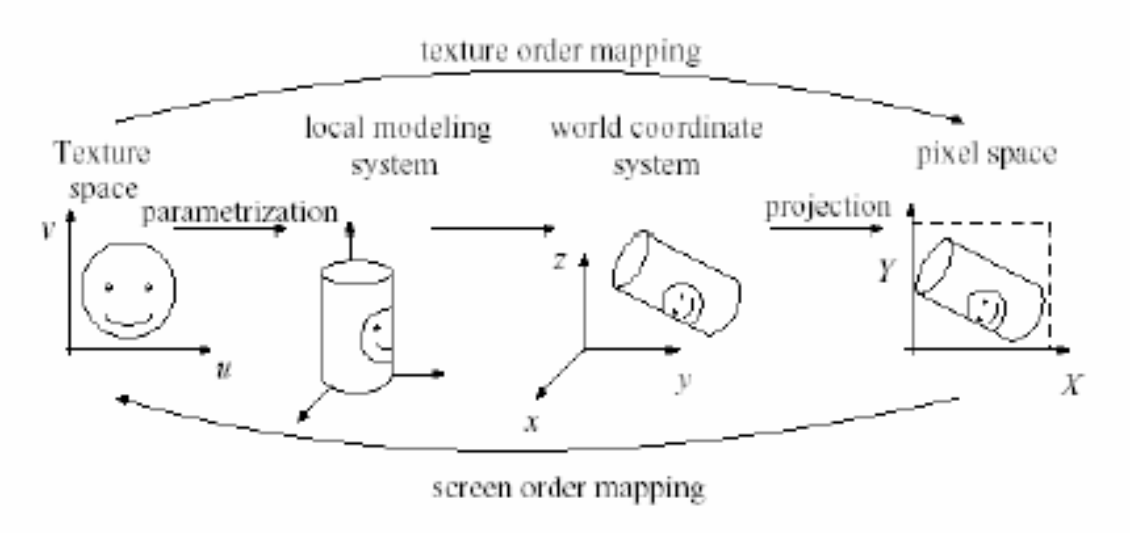
- Nejen difúzní barva, ale i další parametry se dají měnit jinými texturami
  - spekulární barva Phongova odrazu
  - směry normál (bump-mapping) nahrazení složité (mikro) geometrie
  - Příchozí světlo (mapy okolí, nebo reflexní mapy)
  - Náhodné textury (užití syntézi šumu)
  - Fraktální textury (deterministické i stochastické)

9. Jaký sled transformací se uplatní při nanášení 2D textury na povrch tělesa ? Vysvětlete přímé a inverzní mapování při nanášení textur.

Mapování textur <=> warp obrazu



Přímé na inverzní nanášení textur



Přístupy nanášení textur

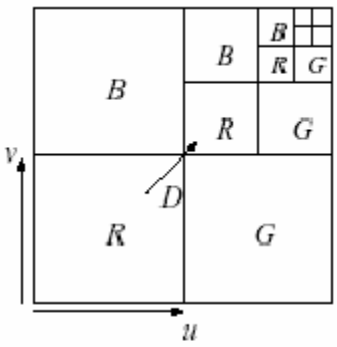
1) prohlížení obrazového prostoru (inverzní mapování)
   for all (x,y)
   (u,v)=f^{-1}(x,y)
   SCREEN[x][y] = TEXTURE[u][v]
2) prohlížení prostoru textury (přímé mapování)
   for all (u,v)
   (x,y) = f(u,v)
   SCREEN[x][y] = TEXTURE[u][v]

10. MIP-MAP metoda aplikace textury.

"multum in parvo"

$MIP[1..M,1..M]$

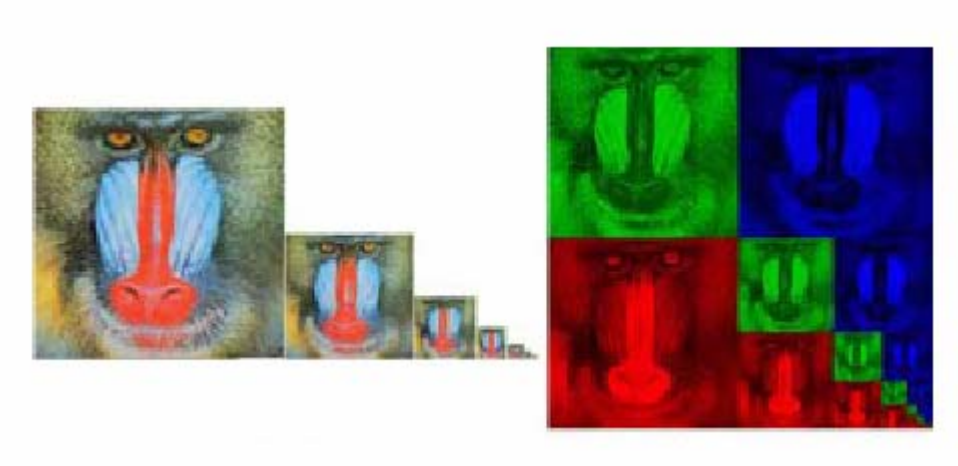
$$1 + 2^{-2} + 2^{-4} + \ldots \approx 1.33$$



$$R(u,v,D) = MIP[(1-2^{-D}).M + u.2^{-D}, (1-2^{-D}).M + v.2^{-D}]$$

$$G(u,v,D) = MIP[(1-2^{-(D+1)}).M + u.2^{-D}, (1-2^{-D}).M + v.2^{-D}]$$

$$B(u,v,D) = MIP[(1-2^{-D}).M + u.2^{-D}, (1-2^{-(D+1)}).M + v.2^{-D}]$$



Aplikace MIP-MAP
Volba **D** podle rozsahu d textury umístěné do plochy pixelu:
Aproximace d:

$$d = \max\left\{\begin{array}{l}\left\|[u(x+1),v(x+1)]-[u(x),v(x)]\right\|,\\ \left\|[u(y+1),v(y+1)]-[u(y),v(y)]\right\|\end{array}\right\}$$

$$\approx \max\left\{\sqrt{\left(\frac{\partial u}{\partial x}\right)^2+\left(\frac{\partial v}{\partial x}\right)^2},\sqrt{\left(\frac{\partial u}{\partial y}\right)^2+\left(\frac{\partial v}{\partial y}\right)^2}\right\}$$

Požadovaná úroveň **D** v piramidě

$$D = \log_2(\max\{d,1\})$$

Aplikace MIP-MAP

Minimum 1 vzplývá ze situace, kdy pixel je pokryt pouze částí texelu. Pak není nutné filtrovat.

$$D = \log_2(\max\{d,1\})$$

Jednoduché použití Trunc/Round pro výpočet indexu na základě **D** by vedl k nespojitostem při zobrazení.
Použijeme interpolaci

$$R_{filtered} = R(u,v,Trunc(D)).(1-Fract(D)) + R(u,v,Trunc(D)+1).Fract(D)$$

$$G_{filtered} = \ldots, \quad B_{filtered} = \ldots$$

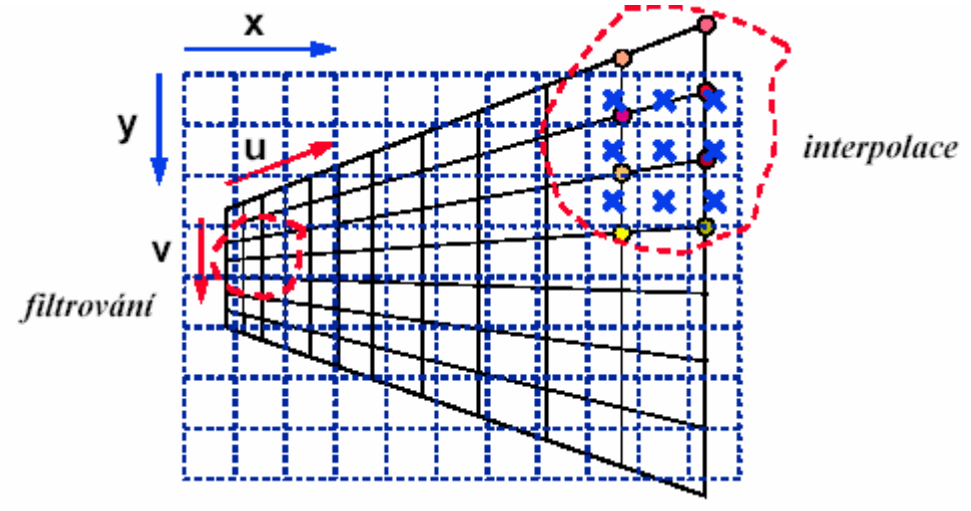
MPG: MIP – ultum in parvo = mnohé v malém

Reprezentace obrazu metodou mip-maping se používá při mapování čtvercových textur. Jeho výhodou je, že v jedné reprezentaci je uchováváno více obrazů v různém rozlišení, což zvyšuje především rychlost mapování takto reprezentované textury.

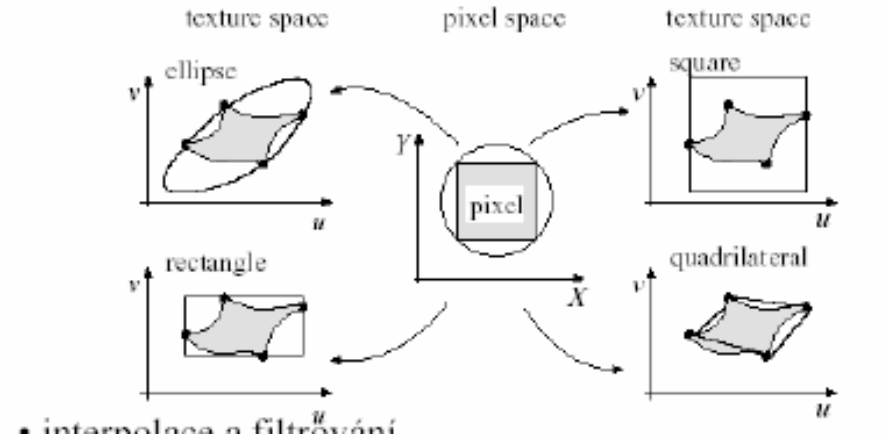
Já: Používá se pro různé úrovně kvality zobrazení textury (vzdálenost texturovaného tělese od pozorovatele)

11. Vysvětlete postup při nanášení textury definované tabulkou.

Textura definovaná tabulkou



Předběžné filtrování textur

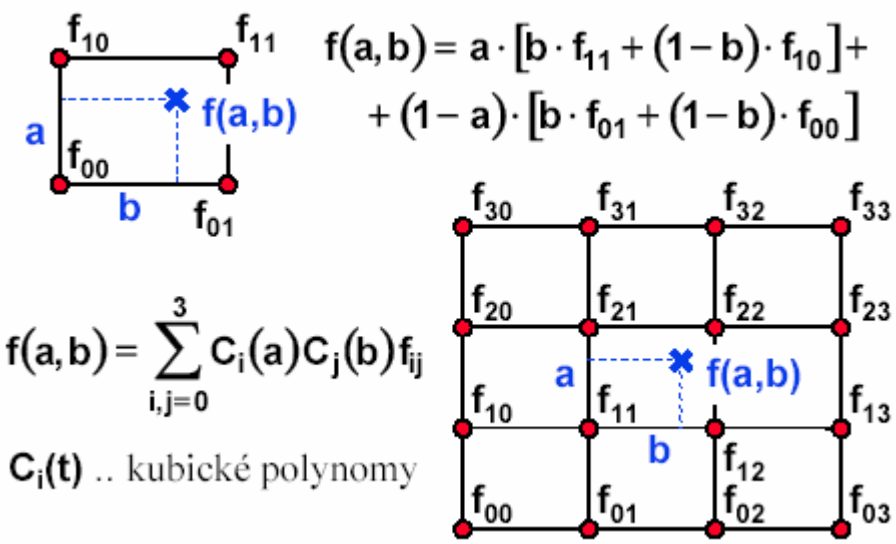


- interpolace a filtrování
- aproximace pokryté plochy obdélníku a filtrování
- MIP-MAP
- Součtová tabulka
- EWA filtrování

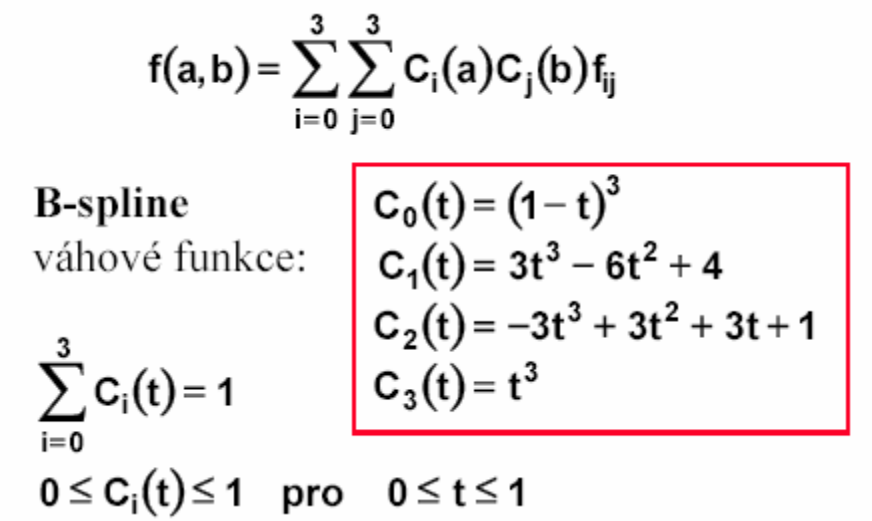
Typy interpolace

- **bez interpolace** (zaokrouhlení)
  - nejjednodušší a nejrychlejší metoda
  - pokud se rozlišení obrázku blíží rozlišení textury vznikají výrazné a nepříjemné artefakty (Doom)
- **bilineární** interpolace
  - zajišťuje **spojitost** obrazové funkce (C^0)
- **polynomiální** interpolace (např. spline funkce)
  - **spojitost** vyšších řádů **(**u bikubické až C^2)
  - výpočetně náročná (kombinace 9-16 hodnot ve 2D)

Bilineární a bikubická interpolace



Kubická B-spline interpolace



12. Vysvětlete důvody a princip nanášení textur přes pomocné povrchy.

Já: Využívá se pro nanášení textury na tělesa se složitým povrchem.

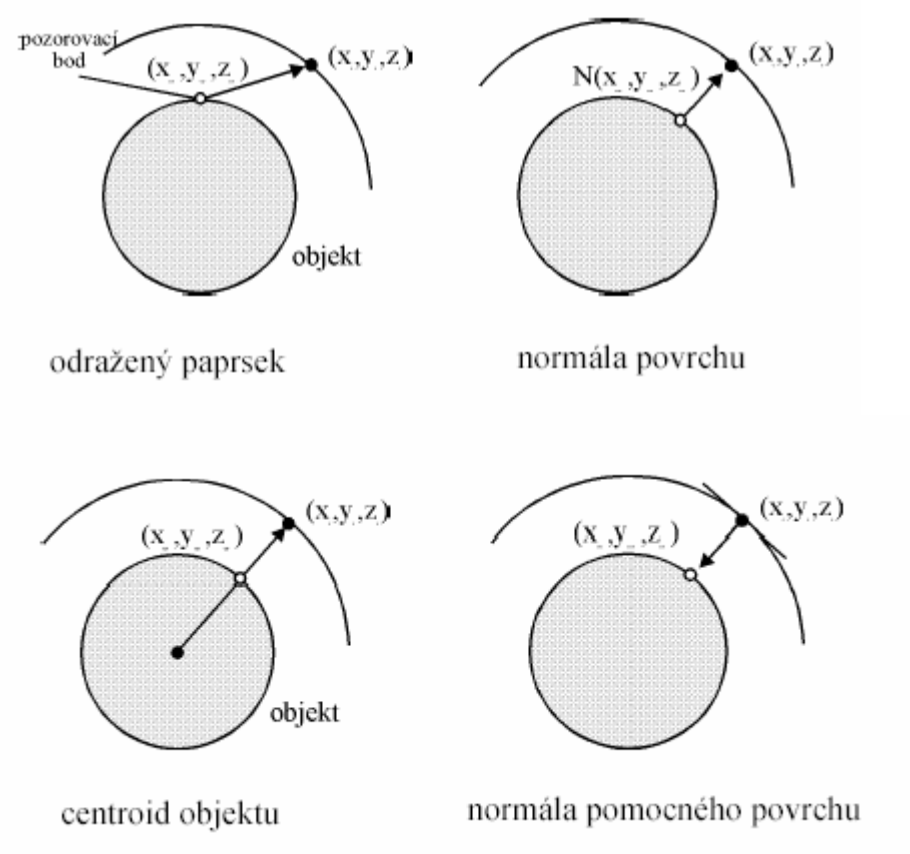
Já: Textura se nanese na pomocný povrch, následně se přenese na povrch původního tělesa, k tomu jsou různé postupy (odražený paprsek(lesklý povrch), normála povrchu, centroid povrchu, normála pomocného povrchu)

2D textury přes pomocné povrchy

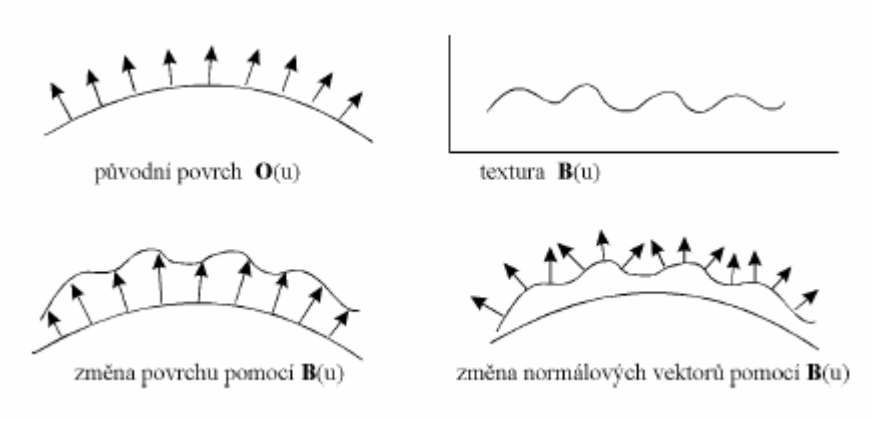
1. 2D textury T(u,v) nanesené na jednoduchý povrch T'
2. Z pomocného povrchu přeneseme texturu na těleso s povrchem O.

Jednoduché pomocné povrchy:
Rovnina, válcová plocha, krychle, koule

Reflexní mapování

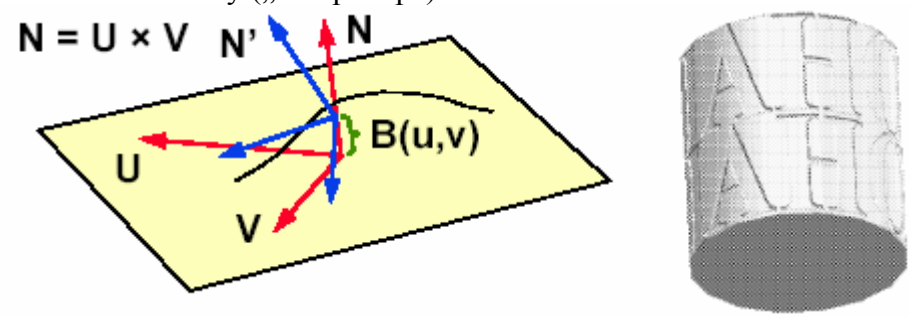


13.Co jsou hrbolaté textury, jak jsou aplikovány ?



Já: Místo zvrásnění povrchu změním jen normálové vektory tak aby odpovídaly zvrásněnému povrchu.

Modulace normály („bump map“)



P’(u,v) = P(u,v) + B(u,v).N/|N|

- napodobení **nerovností** na povrchu tělesa
- **B(u,v)** je funkce lokálního posunutí povrchu:
  + ven z tělesa, - dovnitř tělesa

Modulace normály

Původní normála: $\mathbf{N} = \mathbf{U} \times \mathbf{V}$

Posunutý pod: $$\mathbf{P}'(u,v) = \mathbf{P}(u,v) + \frac{\mathbf{B}(u,v) \cdot \mathbf{N}}{|\mathbf{N}|}$$

Aproximace **modifikované normály:**

$$\mathbf{N}' = \mathbf{N} + \frac{\frac{\partial \mathbf{B}}{\partial u}(u,v) \cdot (\mathbf{N} \times \mathbf{V}) - \frac{\partial \mathbf{B}}{\partial v}(u,v) \cdot (\mathbf{N} \times \mathbf{U})}{|\mathbf{N}|}$$

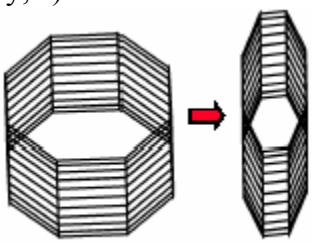
Modelování – deformace

14. Co jsou Barrovy deformace, jaký je výsledek jejich aplikace ? Zapište správný vztah pro transformaci zkroucení okolo osy z.

Barrovy deformace (X, Y, Z,) = F(x, y, z)

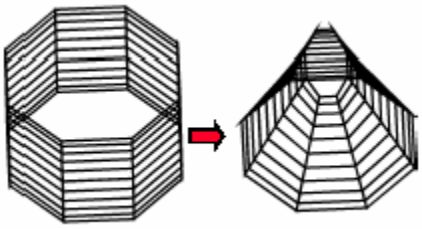
1. Změna měřítka

$$(X, Y, Z) = (s_x.x, s_y.y, s_z.z)$$



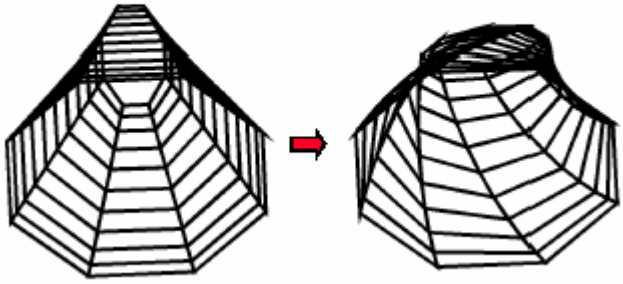
2. Zeslabování („tapering“) ve směru osy z

$$(X, Y, Z) = (r_x x, r_y y, z)$$



3. Zkroucení („twisting“) - okolo osy z

$$(X, Y, Z) = (x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta, z)$$

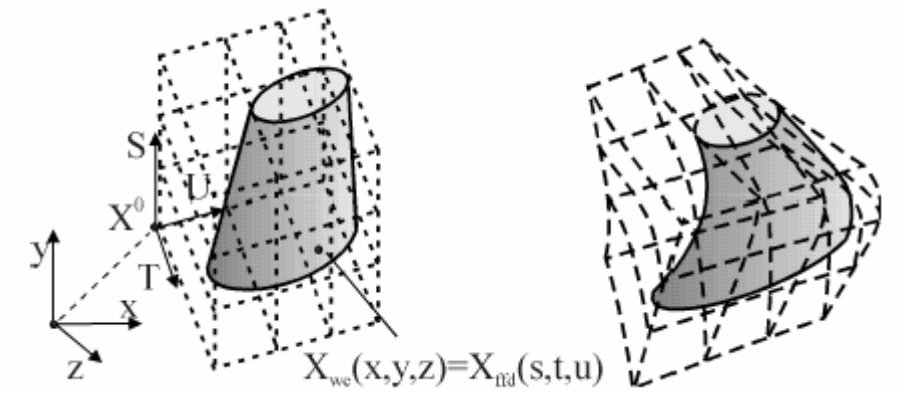


15. Co je FFD ? Vysvětlete princip a účinek. Lze je aplikovat na zvolenou část tělesa? Diskutujte spojitost v deformovaném tělese a změnu objemu.

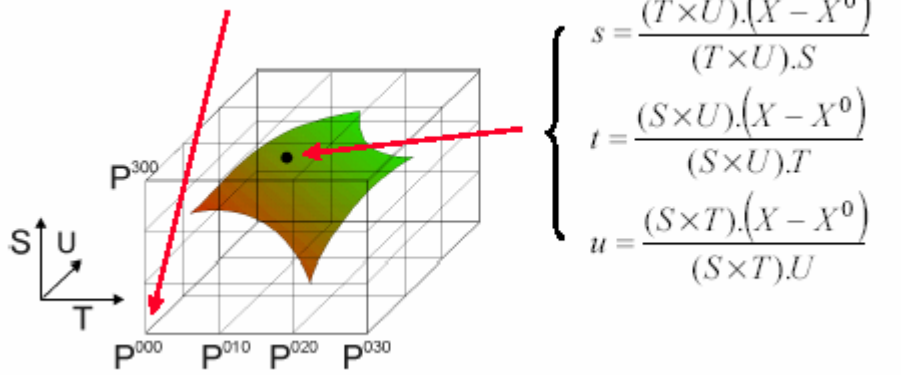
Volné deformace (Sederberg&Parry)

SIGGRAPH: Dobrou fyzikální analogii pro FFD zvážení průzračného paralelního spojení, ohebná plastická hmota ve které je uložen objekt nebo množství objektů, které chceme deformovat. Objekty si představujeme také jako ohebné, takže se deformují podle plastického okolí.

„volné deformace“ – Free Form Deformation – FFD



$$X = X^0 + s.S + t.T + u.U$$



$$s = \frac{(T \times U).(X - X^0)}{(T \times U).S}$$

$$t = \frac{(S \times U).(X - X^0)}{(S \times U).T}$$

$$u = \frac{(S \times T).(X - X^0)}{(S \times T).U}$$

body deformační sítě: $P^{ijk} = X_0 + \frac{i}{l}S + \frac{j}{m}T + \frac{k}{n}U$

$$X_{ffd} = \sum_{i=0}^{l}\binom{l}{i}(1-s)^{l-i}s^i\left[\sum_{j=0}^{m}\binom{m}{j}(1-t)^{m-j}t^j\left[\sum_{k=0}^{n}\binom{n}{k}(1-u)^{n-k}u^k P^{ijk}\right]\right]$$

Spojitost při skládání FFD

$$X_{ffd}(s,t,u) = X_{ffd}(s(v,w),t(v,w),u(v,w))$$

nezávisí na deformaci

derivace složené funkce

$$\frac{\partial X_1(v,w)}{\partial v} = \frac{\partial X_1}{\partial s}\cdot\frac{\partial s}{\partial v} + \frac{\partial X_1}{\partial t}\cdot\frac{\partial t}{\partial v} + \frac{\partial X_1}{\partial u}\cdot\frac{\partial u}{\partial v}$$

$$\frac{\partial X_1(v,w)}{\partial w} = \frac{\partial X_1}{\partial s}\cdot\frac{\partial s}{\partial w} + \frac{\partial X_1}{\partial t}\cdot\frac{\partial t}{\partial w} + \frac{\partial X_1}{\partial u}\cdot\frac{\partial u}{\partial w}$$

Postačující podmínky pro zachování C^1 spojitosti: 2 „sousední" deformace X_1 a X_2 (pro s=0)

$$\frac{\partial X_1(0,t,u)}{\partial s} = \frac{\partial X_2(0,t,u)}{\partial s}$$

$$\frac{\partial X_1(0,t,u)}{\partial t} = \frac{\partial X_2(0,t,u)}{\partial t}$$

$$\frac{\partial X_1(0,t,u)}{\partial u} = \frac{\partial X_2(0,t,u)}{\partial u}$$

Změna objemu při FFD

SIGGRAPH: Další důvod že FFD je tak dobře aplikovatelná na solid modeling je že nám poskytne kontrolu nad změnami objemu které těleso prodělává pod FFD. Změna objemu kterou FFD provede(imposes) nad odlišným elementem je dána Jakobiánem FFD. Zde je daná FFD a její Jakobián

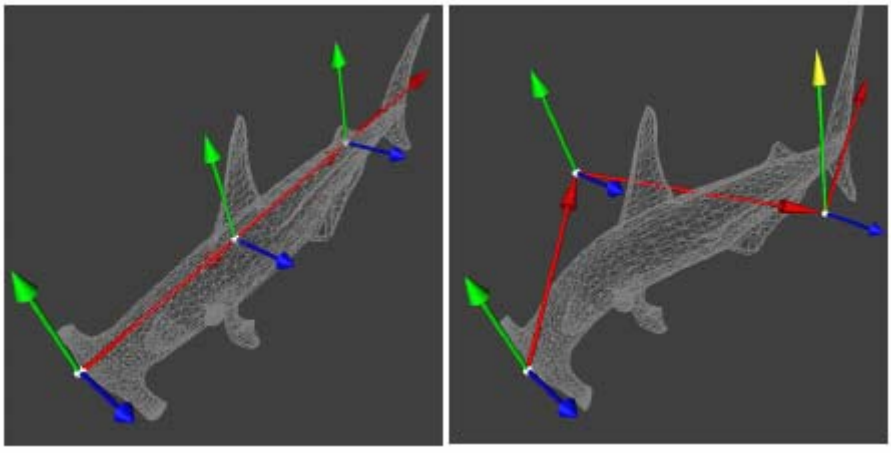
$$FFD(x,y,z) = (F(x,y,z),G(x,y,z),H(x,y,z))$$

$$Jac(FFD) = \begin{vmatrix} \frac{\partial F}{\partial x} & \frac{\partial F}{\partial y} & \frac{\partial F}{\partial z} \\ \frac{\partial G}{\partial x} & \frac{\partial G}{\partial y} & \frac{\partial G}{\partial z} \\ \frac{\partial H}{\partial x} & \frac{\partial H}{\partial y} & \frac{\partial H}{\partial z} \end{vmatrix}$$

Diferenciální objemový element

$$dx.dy.dz \Rightarrow Jac(FFD(x,y,z)).dx.dy.dz$$

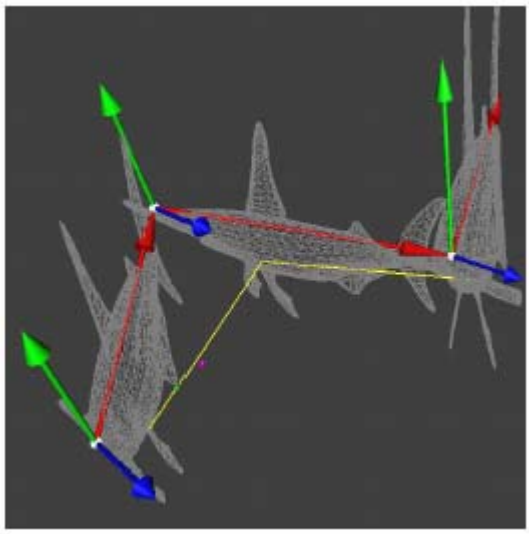
16.Na jakém principu je založena deformace tělesa "deCasteljau" ?



$$\Phi[p,q]\begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} q_x - p_x & s_x & t_x & p_x \\ q_y - p_y & s_y & t_y & p_y \\ q_z - p_z & s_z & t_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

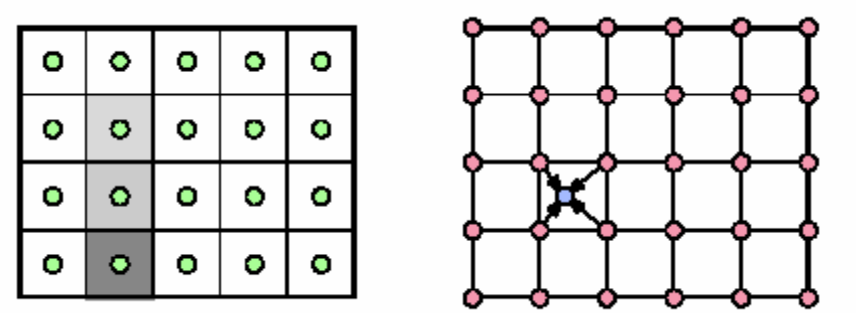
$$P_i^0(u) = P_i \qquad 0 \le i \le n$$

$$P_i^j(u) = \Phi\left[P_i^{j-1}, P_{i+1}^{j-1}\right](u) \qquad 1 \le j \le n,\ 0 \le i \le n-j$$



Zobrazování objemových dat

17. Charakterizujte voxelová a buňková 3D data. Jakým způsobem zjistíte hodnotu v obecném prostorovém bodě objemových dat ?



voxely
(naměřené hodnoty jsou uprostřed)

buňky
(naměřené hodnoty jsou ve vrcholech)

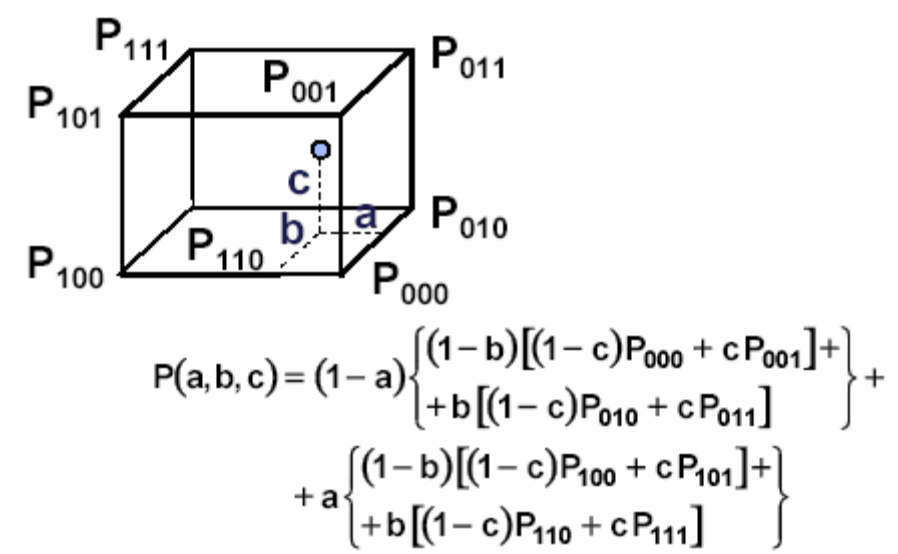
Interpolace v buňkách

- **polynomiální interpolace** a **aproximace**

  - pro topologicky pravidelné mřížky
- **trilineární** interpolace
  - jednoduchý výpočet, není hladká
- **trikvadratická** nebo **trikubická** aproximace
  - hladké, ale vyžadují topologickou pravidelnost
- **radiální aproximace**
  - vhodná i pro topologicky nepravidelné mřížky

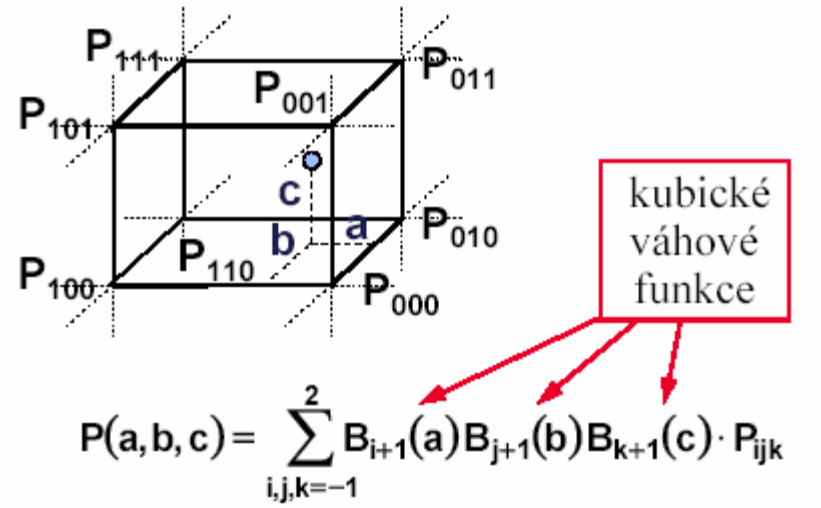
18. Vysvětlete a porovnejte trilineární, trikubickou a radiální aproximaci.

Trilineární interpolace



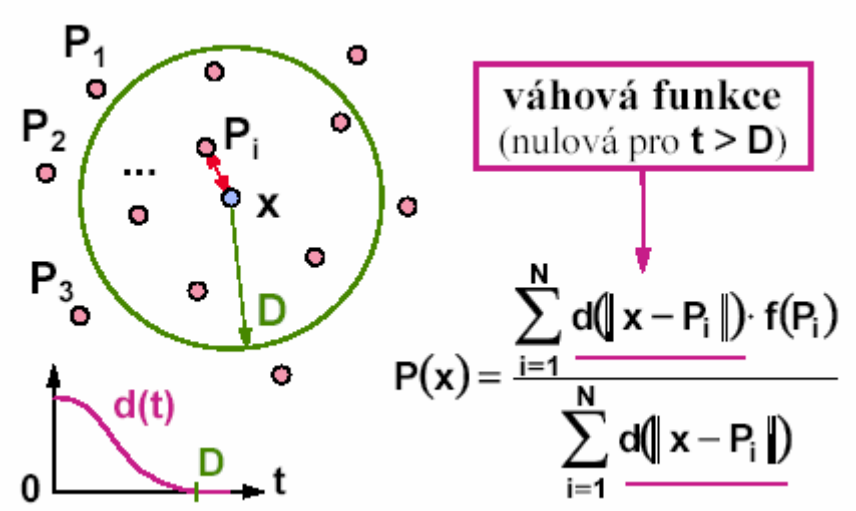
$$P(a,b,c) = (1-a)\left\{\begin{matrix}(1-b)\left[(1-c)P_{000} + cP_{001}\right]+\\ +b\left[(1-c)P_{010} + cP_{011}\right]\end{matrix}\right\} +$$
$$+ a\left\{\begin{matrix}(1-b)\left[(1-c)P_{100} + cP_{101}\right]+\\ +b\left[(1-c)P_{110} + cP_{111}\right]\end{matrix}\right\}$$

Trikubická aproximace



$$P(a,b,c) = \sum_{i,j,k=-1}^{2} B_{i+1}(a)B_{j+1}(b)B_{k+1}(c)\cdot P_{ijk}$$

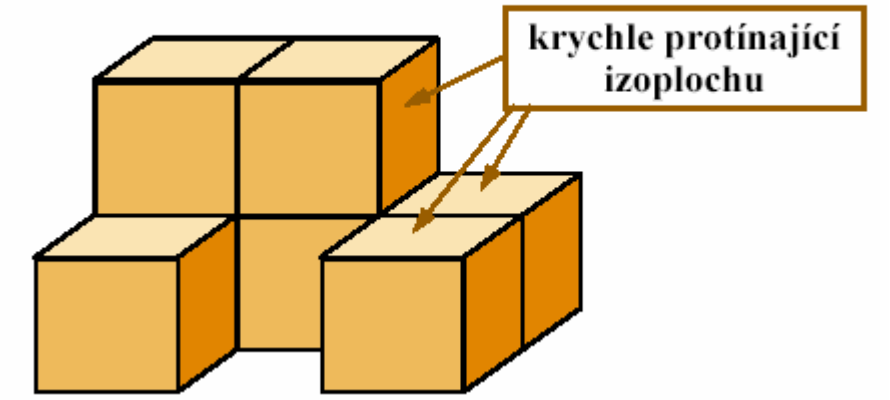
Radiální aproximace



19. Vysvětlete zobrazování objemových dat pomocí tzv. neprůhledných, případně poloprůhledných kostek. Uveďte nedostatky a jejich možné odstranění.

Neprůhledné kostky

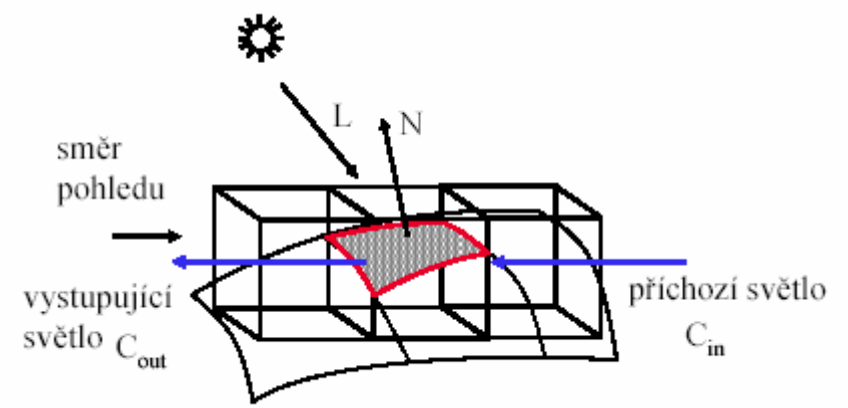
$$\min\{V(P_{ijk})\} \le h < \max\{V(P_{ijk})\}$$



- zobrazení krychlí, které protínají **zadanou izoplochu**
  - podle směru pohledu stačí kreslit pouze tři stěny krychle
- aproximace je příliš **hrubá** (plocha je hranatá)
  - lepší vzhled – **spojité stínování** gradientní metodou (hranaté okraje však zůstávají)
- současné zobrazení **několika izoploch**
  - technika poloprůhledných ploch (kanál alfa)

20. Vysvětlete metodu přímého zobrazování objemových dat (Levoy). Co je vstupem a výstupem algoritmu ?

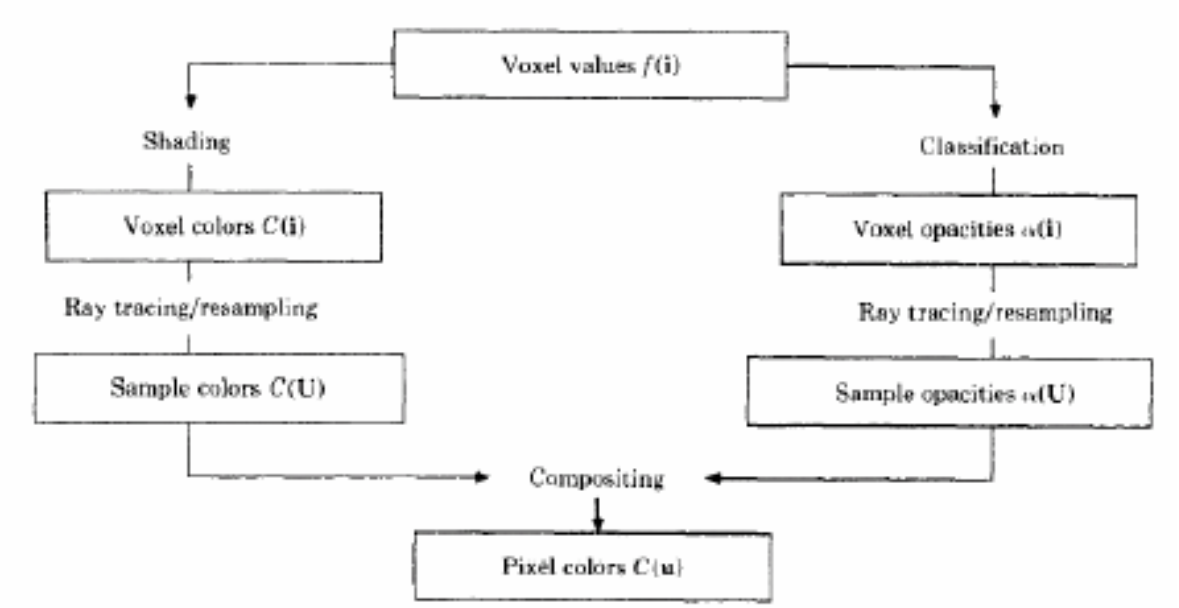
Levoy – direkt volume visual.



Předzpracování vytvoří 2 pomocné objemy:

- pomocné pole barev C(X)
  pro každý voxel vypočte barvu Phongovým modelem
  odhad normály (gradientu) symetrickou diferencí
- pomocné pole neprůhlednosti α(X)

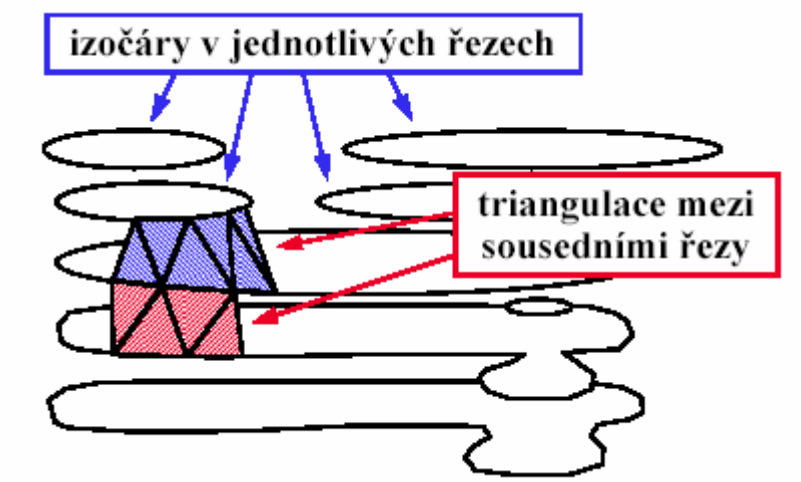
  klasifikace závislá na aplikaci



MPG: Algoritmus má dvě části v první (předspracování) se nastaví pole barev C(X) a pole neprůhledností α(X).
Ve druhém kroku se vytváří výsledný dvojrozměrný obraz na průmětně kombinací obou pomocných objemů C(X) a α(X) rovnoběžným promítáním. Pro každý pixel je do objemu vyslán jeden paprsek. Ten prochází pole C(X) a α(X) a akumuluje hodnoty intenzity (jasu). Paprsek vstupuje do objemu voxelu s intenzitou (jasem) C_{in} je částečně utlumen průchodem jeho objemem o neprůhlednost α a získá část jeho barvy.

21. Jaké jsou hlavní kroky algoritmu pro konstrukci izoplochy pomocí izočar ? (Ekoule et.al). Co je vstupem a výstupem algoritmu ? Řešení konkávních napojení. Větvení 1:M.

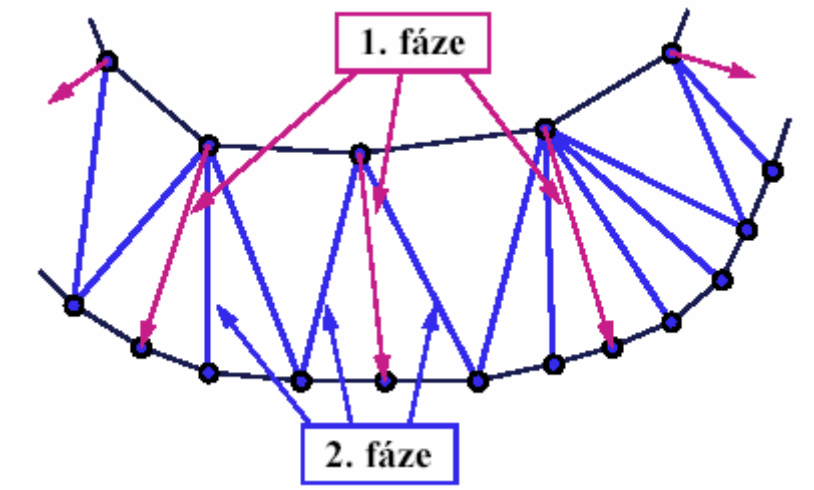
Napojování izočar



- **výpočet izočar** v jednotlivých řezech
  - izočára se reprezentuje jako lomená čára
  - jeden řez může obsahovat několik uzavřených smyček izočar
- **triangulace izoplochy** mezi dvěma řezy
  - trojúhelníky by neměly být příliš protáhlé
  - **topologicky obtížné situace:** několikanásobné větvení (1:N, M:N), nejednoznačné přiřazení napojovaných izočar

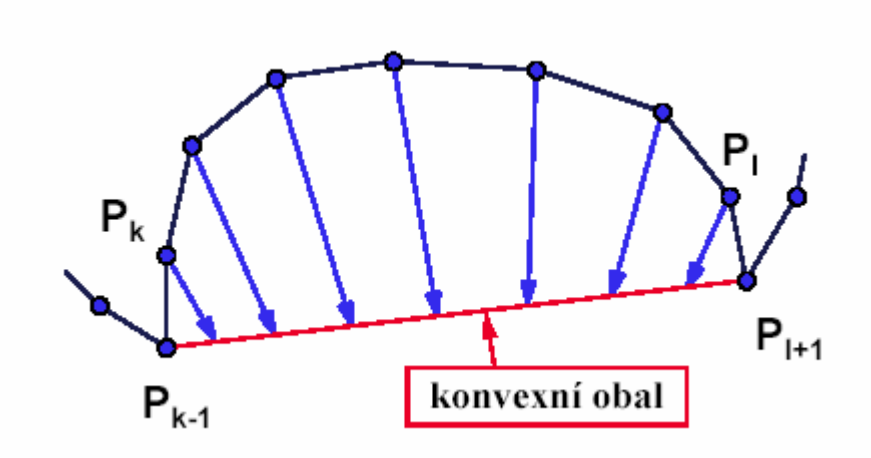
Triangulace
- rozdělení na několik geometricky/topologicky odlišných **případů**:
  - konvexní napojení 1:1
  - nekonvexní napojení 1:1
  - větvení 1:N a M:N

Konvexní napojení 1:1
- pro izočáru s **menším počtem vrcholů**: každý vrchol se spojí s nejbližším bodem naproti
  - efektivní algoritmus pracuje inkrementálně – hledá nejbližšího souseda v okolí naposledy přiřazeného vrcholu
- zbývající vrcholy z **druhé izočáry** se spojí s nejbližšími protějšky
  - některé vrcholy se musejí spojit se dvěma protějšími body (aby vznikla triangulace)

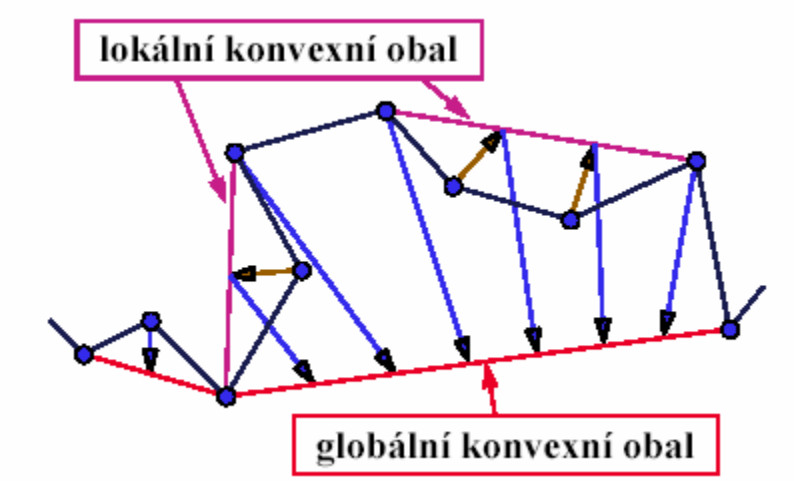


Transformace nekonvexních izočar

- přenášení vrcholů z nekonvexních úseků na **konvexní obal izočáry**
    - hypotéza: konvexní obaly dvou sousedních izočar si budou více podobné
    - přenášením se zachovává pořadí i relativní vzdálenosti vrcholů
- **triangulace** se provádí na konvexních obalech
    - hranami se potom spojí původní vrcholy izočar

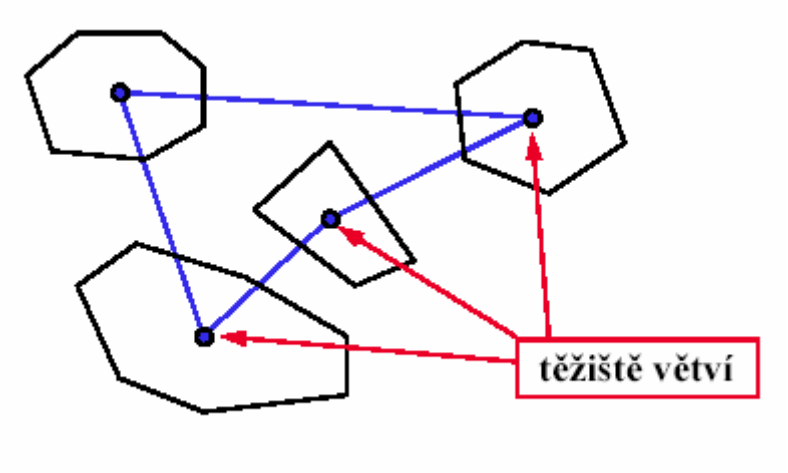


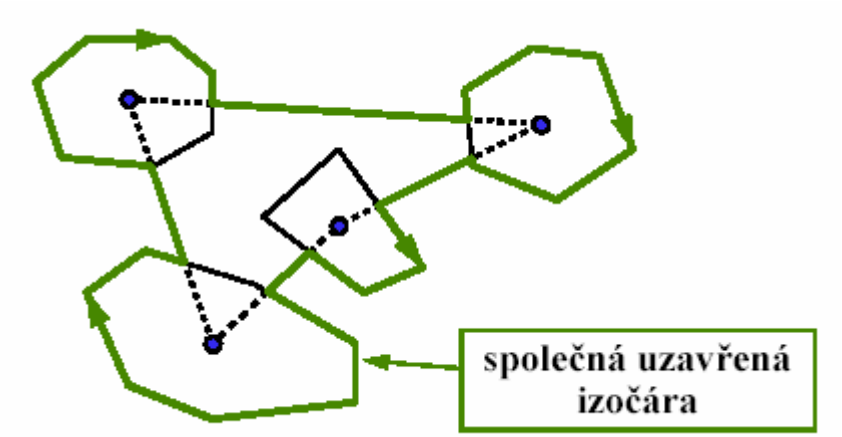
Složitější nekonvexní izočára

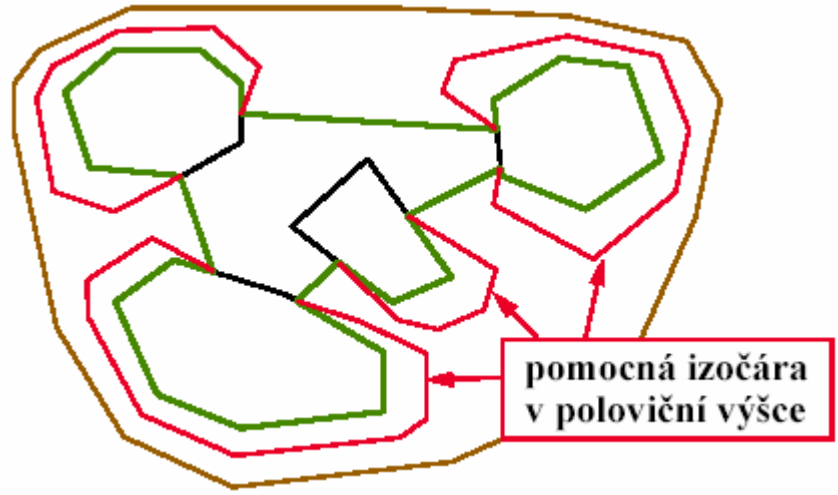


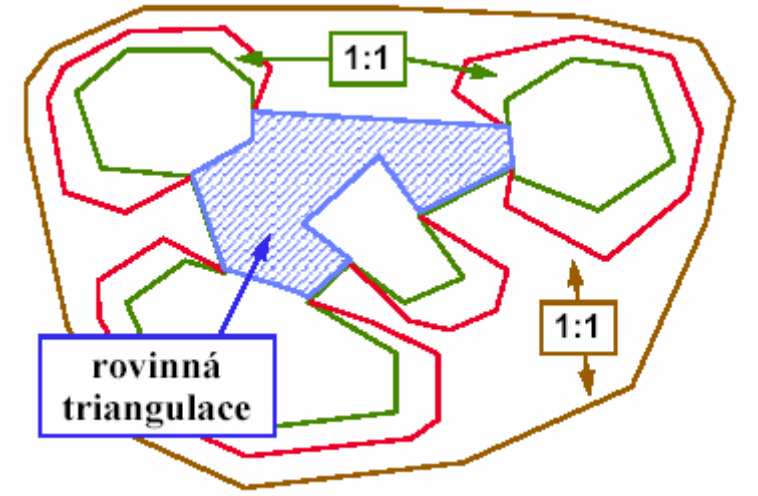
Větvení 1:N

- spočítám **pomocnou izočáru** v polovině mezi oběma řezy
    - konstrukce společné uzavřené izočáry pokrývající všechny spojované větve
    - interpoluje se pomocí prvního kroku algoritmu napojováni 1:1
- N-krát aplikuji napojení 1:1 mezi každou větví a pomocnou izočárou
  + doplním příp. rovinnou triangulaci









22. Na jakém principu pracuje algoritmus „pochodující kostky“ ? Co je vstupem a výstupem algoritmu ? Jak se spočítají souřadnice vrcholů trojúhelníků a normálové vektory ? Nedostatky.

„Pochodující kostky“ (Lorensen)

- **jeden z nejpoužívanějších algoritmů**
  + jednoduchá implementace (i HW)
  + generované trojůhelníky nemají dlouhé hrany
  - velké množství trojúhelníků(i menších než 1 pixel)
  - mohou se objevit drobné chyby v napojení
- generuje **síť trojúhelníků** v každé buňce

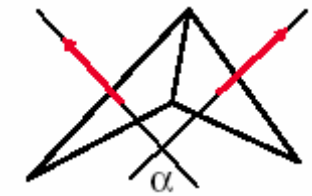
    - topologie – podle konfigurace vrcholů buňky
    - vrchly sítě leží na hranách buňky
- snadno se implementuje **gradientní stínování**

Základní algoritmus
- **4 sousední řezy** se načtou do paměti
- pro každou buňku mezi dvěma prostředními řezy se počítá šíť:
- podle konfigurace hodnot ve vrcholech buňky se z **tabulky** přečte topologie sítě
    - informace o vrcholech na hranách trojúhelníků
- vrcholy sítě se spočítají **lineární interpolací** podle hodnot ve vrcholech buňky
- **normálové vektory** ve vrcholech buňky se spočítají pomocí diferencí a lineárně se pak interpolují do vrcholů sítě
- **síť trojúhelníku** se stínováním se zapíše **na výstup**
    - předpokládá se HW podpora viditelnosti a Gouraudova stínování

Implementace, vylepšení
- spočítané **vrcholy sítě** a **normálové vektory** se ukládají do „cache“ paměti
    - v sousedních buňkách se již nemusí počítat
- náhrada souvislé sítě trojúhelníků v buňce **jedním mnohoúhelníkem** (nerovinným)
    - průmět buňky je často malý (řádově několik pixelů)
    - větší efektivita při kreslení
- **množinové operace**, řezy rovinou
    - paralelně se provádí výpočet několika izoploch

23. Zjednodušení trojúhelníkové sítě. Princip algoritmu s metrikou QEM.

Zjednodušení sítě



Významné hrany – test prostorového úhlu podle prahové hodnoty

Vrchol s právě 2 významnými hranami – **vrchol vnitřní hrany**

Kritérium vzdálenosti od hrany

Vrchol s jednou nebo více než 2 významnými hranami – **roh**

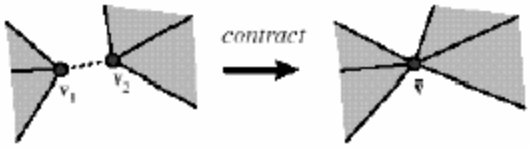
Ponecháváme beze změny

Zjednodušení s metrikou QEM

Surface Simplification Using Quadric Error Metrics
M.Garland, P.Heckbert, CMU, SIGGRAPH 97

$(v_1, v_2) \rightarrow \bar{v}$ párová kontrakce (dvojice vrcholů)

kontrakce existující hrany



kontrakce 2 vrcholů nepropojených hranou

Výběr párů pro kontrakce

Páry vrcholů pro budoucí kontrakce jsou vybrány předem v inicializační fázi

(v_1,v_2) je *platný pár* pro kontrakci, pokud

1. (v_1,v_2) je hrana, *nebo*
2. $||v_1,v_2||<t$, t je zvolený práh

Algoritmus zjednodušování povrchu s QEM
1. Vypočti matice Q pro všechny iniciální vrcholy
2. Vyber všechny platné páry.
3. ¨vypočti optimální kontrakční cíl **v** pro každý platný pár (v_1,v_2). Chyba $\mathbf{v}^T(\mathbf{Q_1}+\mathbf{Q_2})\mathbf{v}$ je *cennou* kontrakce tohoto páru.
4. Všechny páry dej na haldu uspořádaně podle ceny, pár s nejmenší cenou kontrakce dej na vrchol.
5. Opakovaně odeber pár s nejmenší cenou $(\mathbf{v_1},\mathbf{v_2})$ z hlady, kontrahuj tento pár a oprav všechny ceny u párů obsahujících vrchol $\mathbf{v_1}$.

rovina $ax + by + cz + d = 0 \;;\quad a^2 + b^2 + c^2 = 1$

$$\Delta(v) = ([v_x v_y v_z 1]^T) = \sum_{p \in roviny(v)} (p^T v)^2 =$$

$$= \sum_{p \in roviny(v)} v^T (pp^T) v = v^T \left(\sum_{p \in roviny(v)} K_p \right) v$$

matice Q;
výchozí chybová metrika

$$K_p = pp^T = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$

Ve výchozím modelu má každý vrchol sdružené roviny a odpovídající matice Q; matice metrik se přenášejí po kontrakci jako součet $Q=Q_1 + Q_2$

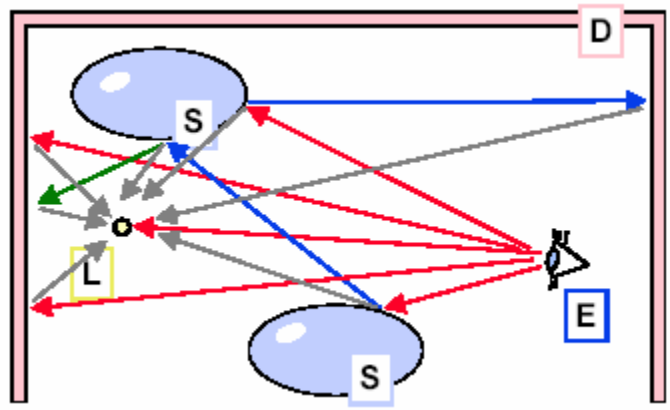
Urychlovací metody pro RT(raytracing)

24. Zformulujte základní úlohu RT, vysvětlete jednotlivé části výpočtu. Naznačte možnosti zrychlení (úspory) výpočtu.

Zobrazovací metody (RT)

- **Rekurzivní sledování paprsku**
  (eye ray-tracing, forward ray-tracing, visibility tracing)
- **BRDF** udává váhu sekundárního paprsku
  - Lesklé plochy: zrcadlový paprsek má obvykle největší váhu, a proto ho sleduji dále
  - Difusní plochy: vše má stejný příspěvek – co dál???
- Nepokryté cesty nahradím **amgientní složkou**
- První lesklý odraz se počítá přesně, ostatní se nahrazují ideálním zrcadlovým odrazem
- **$L[D|S]S_M^*E$**

Sledování paprsku (ray-tracing)

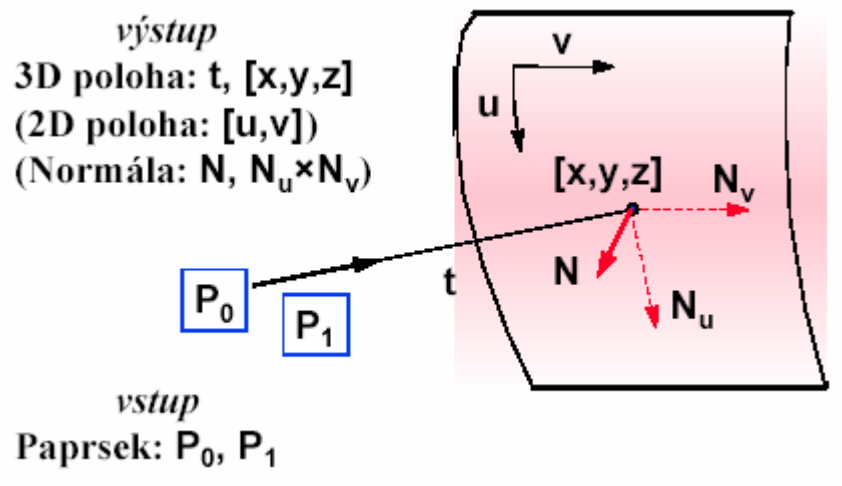


Klasifikace urychlovacích metod

- urychlení výpočtu **„paprsek x scéna“**
  - urychlení testu **„paprsek x těleso“**
    - obalová tělesa, efektivní algoritmy výpočtu průsečíků
  - menší počet testů **„paprsek x těleso**“
    - hierarchie obalových těles, dělení prostoru (prostorové adresáře), směrové techniky (+2D adresáře)
- menší **počet testovaných paprsků**
  - dynamické řízení rekurze, adaptivní vyhlazování
- **zobecněné paprsky** (dávají více informace)
  - polygonální svazek paprsků, kužel, …

25. Při nalezení průsečíku paprsku s povrchem tělesa nás zajímá celý soubor údajů. O které údaje se jedná, jaký je jejich význam a následné využití ?

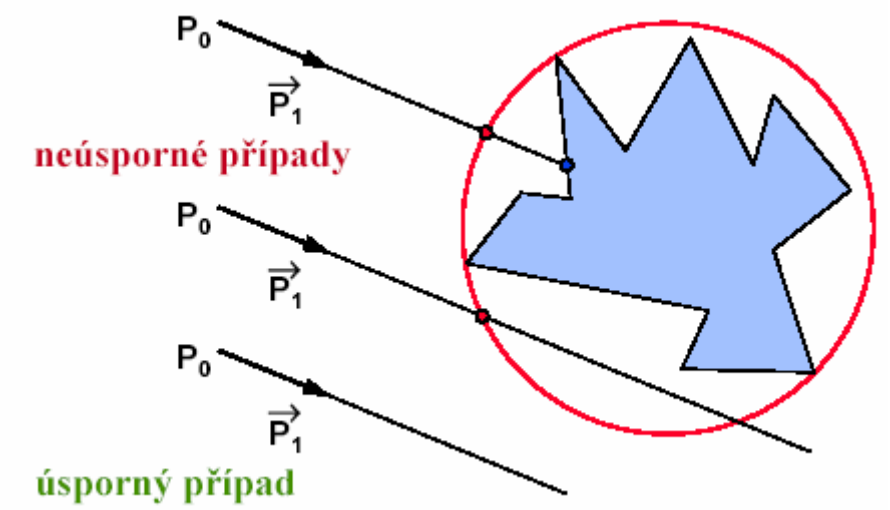
Průsečík paprsku s tělesem



Průsečík paprsku s 3D scénou

- spotřebuje **většinu strojového času**
- scéna je složena z **elementárních těles**
  - koule, kvádr, válec, kužel, jehlan, polygon, …
  - elementární tělesa v CSG
  - počet elementárních těles … N
- klasický algoritmus testuje **každý paprsek** (do hloubky rekurze H) s **každým elementárním tělesem**
  - **O(N)** testů pro jeden paprsek

26. Uveďte několik příkladů obalových těles a vysvětlete jejich použití. Vysvětlete použití hierarchie obalových těles v RT a její přínos. Jak lze charakterizovat efektivitu použití hierarchie obalů ?



Obalové těleso

1. výpočet průsečíku je **jednodušší** než u původního tělesa
   - koule, kvádr v obecné nebo osově rovnoběžné poloze, průnik pásů, …

2. obal by měl **co nejtěsněji obklopovat** původní těleso (pro maximální urychlení)
3. **efektivita** obalového tělesa záleží na vhodném kompromisu 1 a 2
    - celková asymptotická složitost zůstává **O(N)**

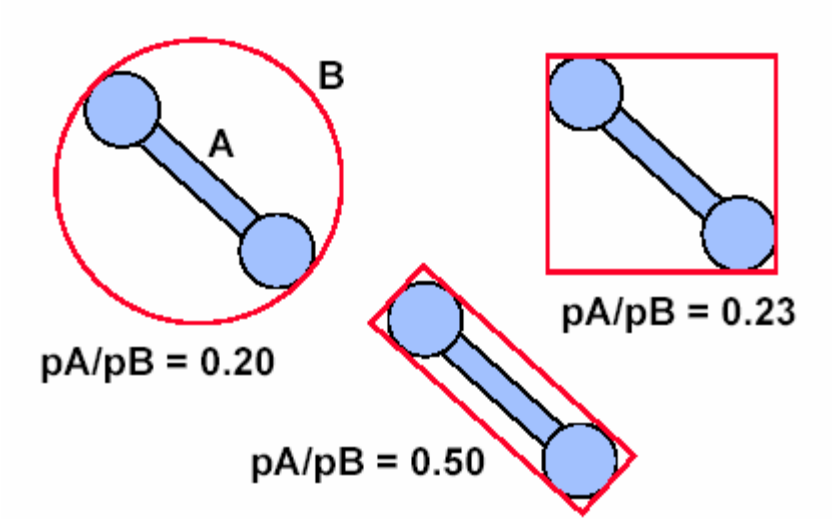
Efektivita obalového tělesa

Očekávaný **průměrný čas výpočtu** průsečíku paprsku s tělesem

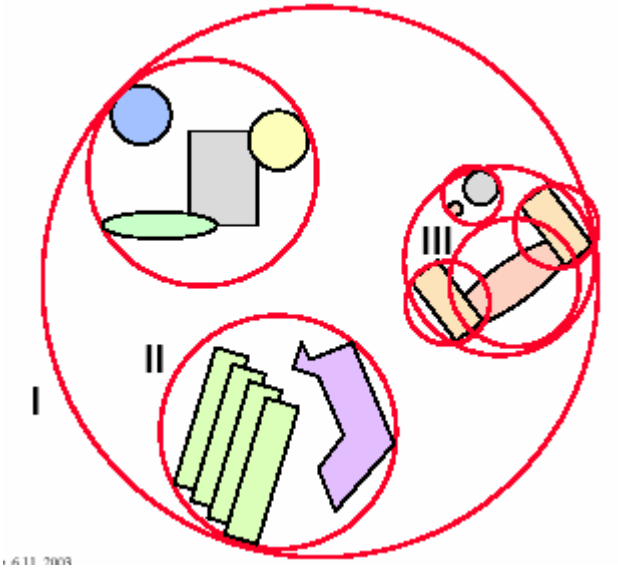
B + p . I < I

I … čas výpočtu průsečíku s **původním tělesem**
B …čas výpočtu průsečíku s **obalovým tělesem**
p … pravděpodobnost zásahu obalového tělesa paprskem (kolik % paprsků protne obalové těleso)

Různě efektivní obaly



Hierarchie obalových těles



Hierarchie obalových těles

- v ideálním případě **snižuje asymptotickou složitost** na **O(log N)**
- vyplatí se zejména u **dobře strukturovaných scén**
    - množství dobře oddělených malých objektů
    - přirozená implementace v CSG reprezentaci (prořezávání CSG strom)
- možnost **automatické konstrukce** hierarchi

Efektivita hierarchie

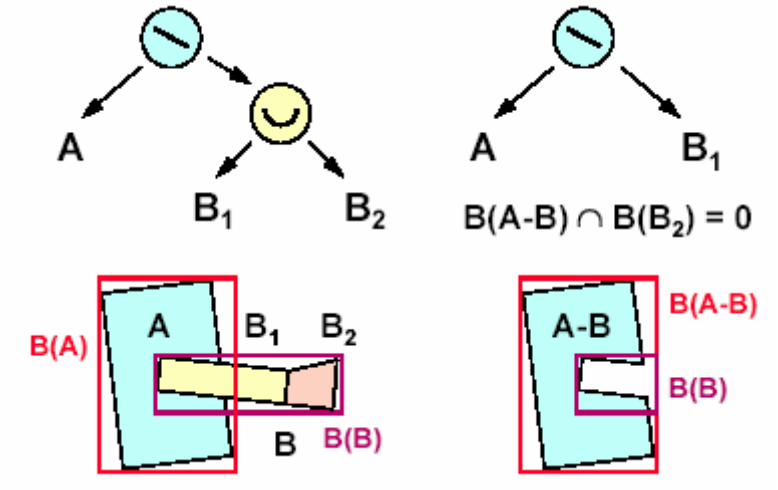
$$K \cdot B + \sum_{i=1}^{K} p_i \, l_i \;\; ? \;\; < \;\; \sum_{i=1}^{K} l_i$$

B … čas výpočtu průsečíku s obalovým tělesem
p_i … pravděpodobnost zásahu i-tého obalového tělesa
I_i … čas výpočtu pro objekty uzavřené v i-tém obalovém tělese

27. Co je „prořezávání CSG stromu“ ? Jak lze tuto metodu aplikovat na obalová tělesa ?

Prořezávání CSG stromu

- efektivní především pro **subtraktivní množinové operace** (průnik, rozdíl)
- primární obalová tělesa jsou přiřazena (omezeným) **elementárním tělesům**
  - velikost se většinou určuje analyticky
- obalová tělesa se pomocí množinových operací **propagují směrem ke kořeni**
- u argumentů **subtraktivních operací** se mohou obalová tělesa **zmenšovat**



28. Jaké výhody přináší při sledování paprsků dělení prostoru zobrazované scény ? Uveďte příklady, stručně charakterizujte jejich podstatu.

Dělení prostoru – prostor adresáře

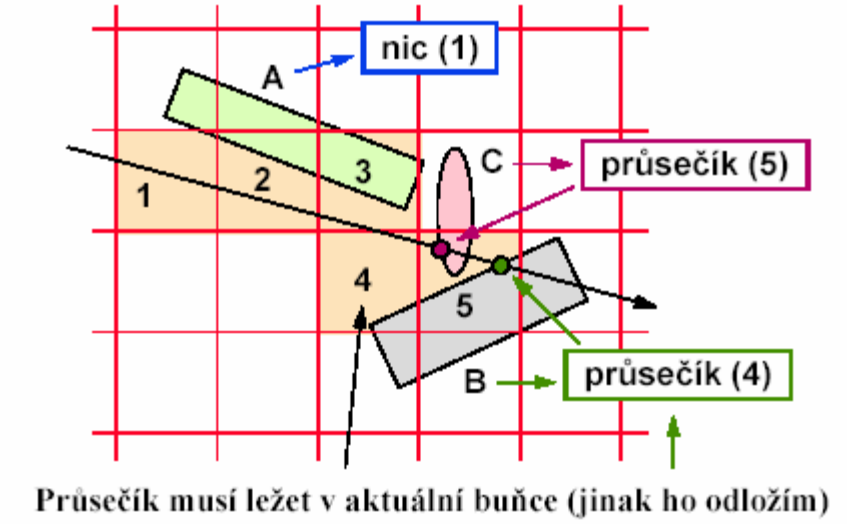
- **uniformní dělení** (stejně velké buňky)
  + jednoduchý průchod
  - mnoho kroků výpočtu
  - velký objem dat
- **neuniformní dělení** (většinou adaptivní)
  + méně kroků výpočtu
  + menší objem dat
  - složitější implementace datové struktury i algoritmu procházení

29. Vysvětlete princip „poštovních schránek“ při sledování paprsku. Uveďte příklad.

Průchod adaptivními strukturami

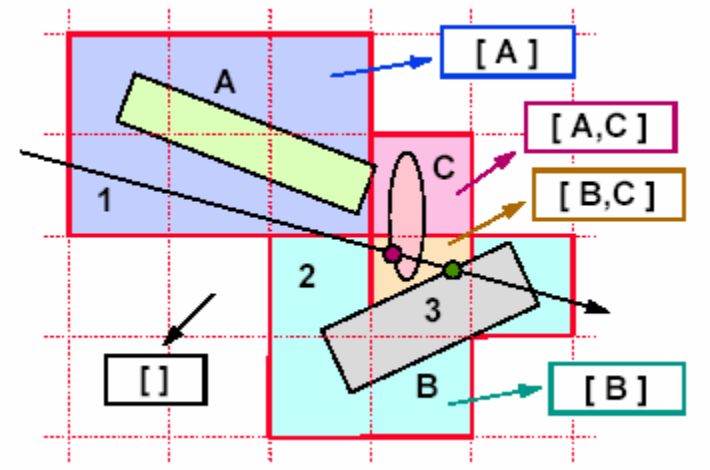
- posunuji se po paprsku a hledám sousední buňku vždy až **od kořene**
- **přípravná fáze:** průchod stromem a rozdělení paprsku na intervaly
  - intervaly parametru **t** přiřazené jednotlivým buňkám, kterými budu procházet
- **pomocné údaje** v datových strukturách (á la „finger tree“)
  - ukazatele na sousední buňky (na stejné úrovni ve stromu), …

Poštovní schránka



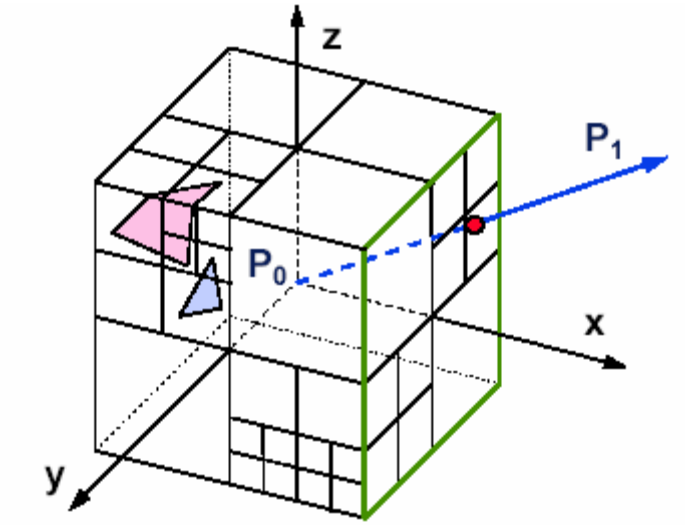
Abstraktní dělení prostoru

- **není třeba testovat** (ani procházet!) seznamy, které jsem již testoval
- seznam musím procházet až v takové buňce, do které zasahuje **jiná** (větší) **množina těles**
- buňky mohou **sdílet shodné seznamy těles**
  - otestované seznamy **označují** zvláštním příznakem
  - procházím pouze **neoznačení** seznamy
  - na úrovni těles používám techniku schránek



30. Princip zrychlení výpočtu stínovacích paprsků pomocí světelné krychle.

Směrové urychlovací techniky
- metody využívající **směrové krychle**:
- **světelný buffer**
  - urychluje stínovací paprsky k bodovým zdrojům
- **koherence paprsků**
  - urychluje všechny sekundární paprsky
- **5D klasifikace paprsků**
- **adresář v průmětně** (předvýpočet viditelnosti)
  - urychluje pouze primární paprsky



Směrová krychle

- **orientovaná** rovnoběžně s osami **x, y, z**
- jednotlivé stěny jsou rozděleny na **buňky**
  - uniformní nebo adaptivní dělení
  - každá buňka obsahuje seznam relevantních objektů (mohou být navíc setříděny vzestupně podle vzdálenosti od středu krychle)
- při uniformním dělení lze pro urychlení využít **HW výpočtu viditelnosti** (z-buffer)

Globální osvětlování

31. Radiometrické veličiny: zářivý tok, zářivost, záře, ozáření, radiozita.

Radiometrické veličiny – Energie, Tok, Ozáření (Irradiance), Radiozita

- Množství energie přijaté (emitované) nějakou části plochy: $\mathbf{Q_{in}(Q_{out})}$ **[Joule=J]**
- **Zářivý tok (výkon)** procházející (emitovaný) nějakou částí plochy za jednotku času: $\mathbf{\phi_{in}(\phi_{out})}$ **[J/sec=Watt]**
- **Ozáření E:** množství energie dopadající na jednotku plochy za jednotku času **[W/m^2]**

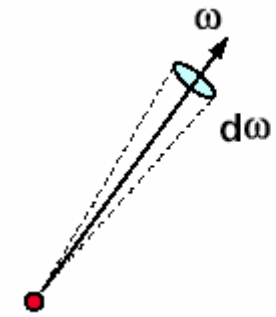
$$E = \frac{d\Phi}{dA}$$

- **Radiozita B:** množství energie vyzářené z jednotkové plochy za jednotku času **[W/m^2]**

$$B = \frac{d\Phi}{dA}$$

Zářivost (Radiant intensity)

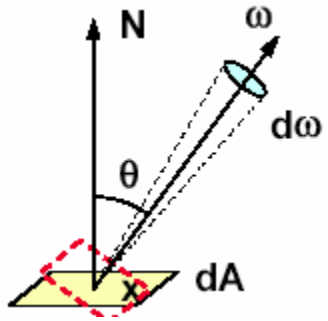
- **Zářivost I:** množství energie vyzáření z bodového zdroje do jednotkového prostorového úhlu za jednotku času [**W/sr**]



$$I = \frac{d\Phi}{d\omega}$$

Záře (Radiance)

- **Záře L:** množství energie dopadající (emitované) z jednotkového prostorového úhlů na jednotkovou plochu kolmou na směr záření **[W/(m^{2}sr)]**

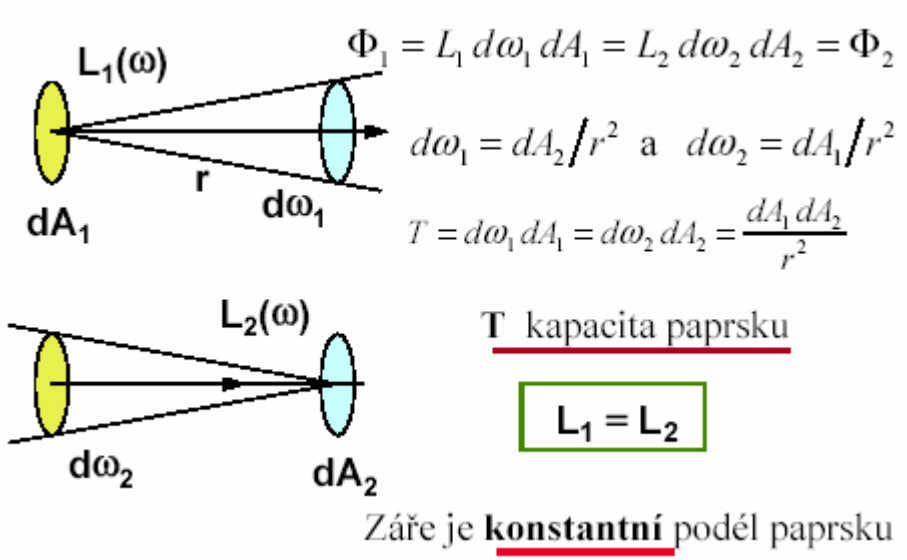


$$L(x,\vec{\omega}) = \frac{d^2\Phi}{dA\, d\omega \cos\theta} = \frac{dB}{d\omega \cos\theta} = \frac{dI}{dA \cos\theta}$$

Symbol	Veličina [cs]	Veličina [an]	Jednotky
Q	Energie	Energy	Joule
Φ	Zářivý tok	Radiant power	Watt
E	Ozáření	Irradiance	W/m^2
B	Radiozita	Radiosity	W/m^2
I	Zářivost	Radiant intensity	W/sr
L	Zář	Radiance	W/(m^{2}sr)

32. Princip zachování záře podél paprsku.

Zachování záře podél paprsku



$$\Phi_1 = L_1\, d\omega_1\, dA_1 = L_2\, d\omega_2\, dA_2 = \Phi_2$$

$$d\omega_1 = dA_2/r^2 \quad \text{a} \quad d\omega_2 = dA_1/r^2$$

$$T = d\omega_1\, dA_1 = d\omega_2\, dA_2 = \frac{dA_1\, dA_2}{r^2}$$

T kapacita paprsku

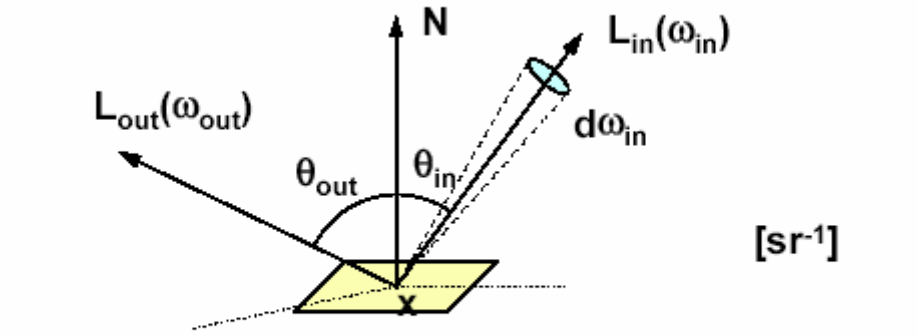
$L_1 = L_2$

Záře je **konstantní** podél paprsku

33. Obousměrná odrazová distribuční funkce - vysvětlete pomocí obrázku. Proč se nazývá "obousměrná" ? Příklady BRDF pro ideální difusní a zrcadlové povrchy.

Dvousměnová odrazová distribuční funkce – BRDF

(**B**idirectional **R**eflectance **D**istribution **F**unction)



$$f_r(x, \vec{\omega}_{in} \rightarrow \vec{\omega}_{out}) = \frac{dL_{out}(x, \vec{\omega}_{out})}{dE(x, \vec{\omega}_{in})} = \frac{dL_{out}(x, \vec{\omega}_{out})}{L_{in}(x, \vec{\omega}_{in}) \cos\theta_{in}\, d\omega_{in}}$$

Vlastnosti BRDF

- Helmholtzův zákon – pro reálné povrchy těles platí:

$$f_r(x, \vec{\omega}_{in} \rightarrow \vec{\omega}_{out}) = f_r(x, \vec{\omega}_{out} \rightarrow \vec{\omega}_{in})$$

- **BRDF** nemusí být obecně isotropní (invariantní k otočení kolem simetrány), např. kovové povrchy leštěné v jednom směru

$$f_r(x, \theta_{in}, \varphi_{in}, \theta_{out}, \varphi_{out}) \neq f_r(x, \theta_{in}, \varphi_{in} + \alpha, \theta_{out}, \varphi_{out} + \alpha)$$

- BRDF je **normalizovaná** – množství odražené energie bude menší než (nebo rovno) množství dopadající energie
- BRDF je **lineární** vzhledem k záření – energie přicházející z různých směrů přispívá nezávisle na výslednou odraženou energii
- Jednotkou je [sr^{-1}]

BRDF příklady

- pro **ideálně difusní** povrchy je BRDF konstantní.

$$f_{r,d} = const$$

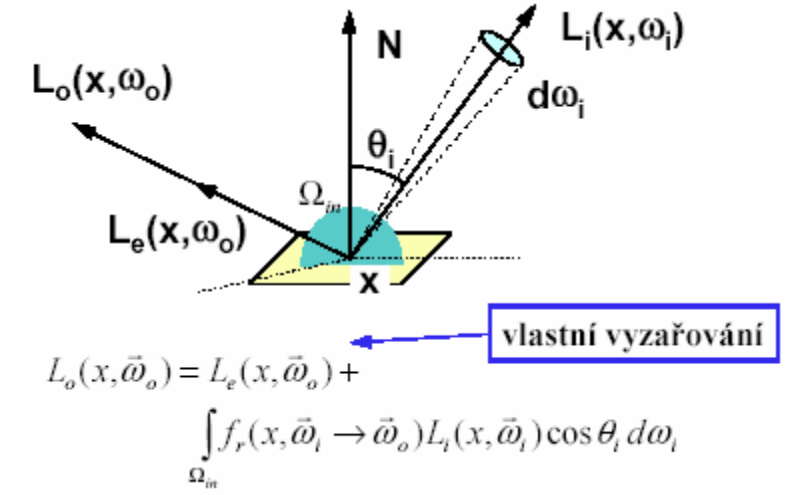
- Pro **ideálně zrcadlové** povrchy je BRDF rovno:

$$f_{r,s}((\theta_i,\varphi_i) \rightarrow (\theta_o,\varphi_o)) = \frac{\delta(\cos\theta_i - \cos\theta_o)}{\cos\theta_o} \cdot \delta(\varphi_i - (\varphi_o \pm \pi))$$

- ve skutečnosti zastoupeny obje složky, tzv. **lesklé** povrchy.

34. Co vyjadřuje rovnice VTIGRE (vacuum time invariant gray radiance equation)?

Zobrazovací rovnice (VTIGRE)



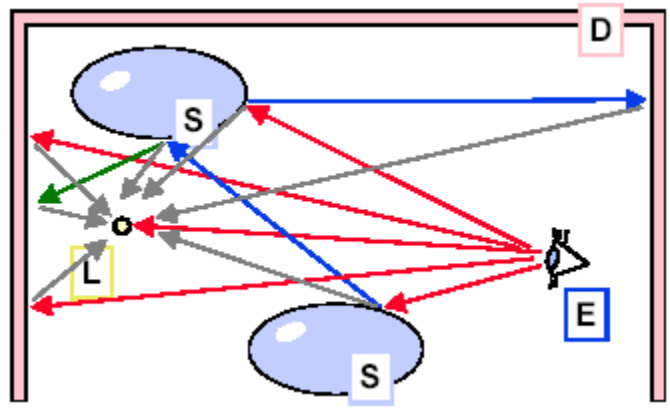
$$L_o(x,\vec{\omega}_o) = L_e(x,\vec{\omega}_o) + \int_{\Omega_{in}} f_r(x,\vec{\omega}_i \rightarrow \vec{\omega}_o) L_i(x,\vec{\omega}_i) \cos\theta_i \, d\omega_i$$

35. Pomocí obrázků a operátorů šíření světla charakterizujte jednotlivé metody výpočtu osvětlení: RT, radiozita, obousměrné sledování cest, kombinaci radiozity a RT (dvoukroková metoda).

Zobrazovací metody (RT)

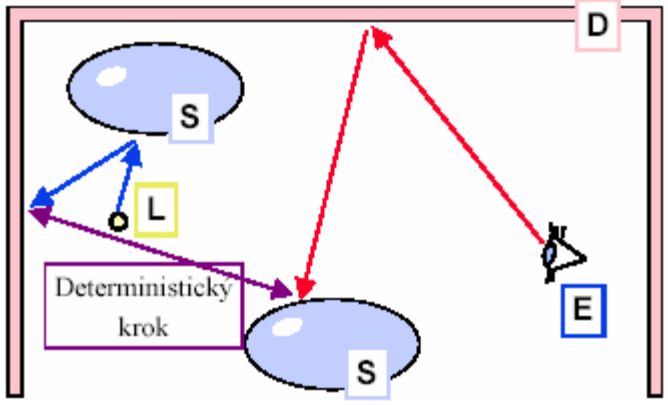
- **Rekurzivní sledování paprsku**
  (eye ray-tracing, forward ray-tracing, visibility tracing)
- **BRDF** udává váhu sekundárního paprsku
  - Lesklé plochy: zrcadlový paprsek má obvykle největší váhu, a proto ho sleduji dále
  - Difusní plochy: vše má stejný příspěvek – co dál???
- Nepokryté cesty nahradím **amgientní složkou**
- První lesklý odraz se počítá přesně, ostatní se nahrazují ideálním zrcadlovým odrazem
- **L[D|S]S_M^*E**

Sledování paprsku (ray-tracing)



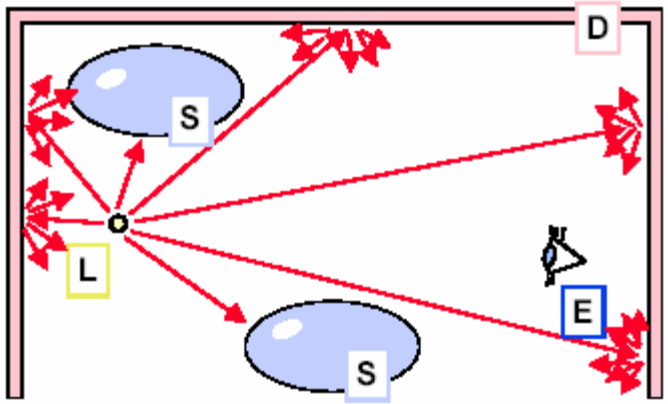
Obousměrné sledování cest

- **Obousměrné sledování cest** kombinuje sledování cest a sledování světla.
- Po několika náhodných krocích ze zdroje a od pozorovatele se provede **deterministický** krok, který spojí konce (nebo vnitřní body) obou cest.
- Lepší konvergence než u předcházejících metod.



Radiozitní metoda

- vícenásobné difusní odrazy
- nerozlišuje primární a sekundární zářiče
- řešení metodou konečných prvků
- závisí na kvalitě sítě ploch a elementů
- pohledově nezávislé řešení
- světelné cesty: **LD*E**



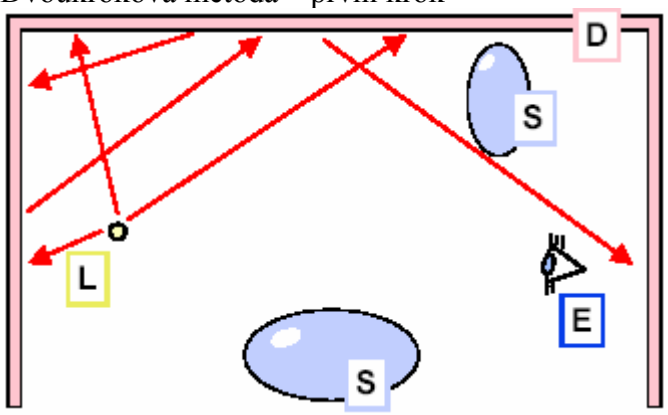
Jednoduchá dvoukroková metoda

- klasická radiozitní metoda: **LD***
  - výpočet osvětlení na všech ploškách scény

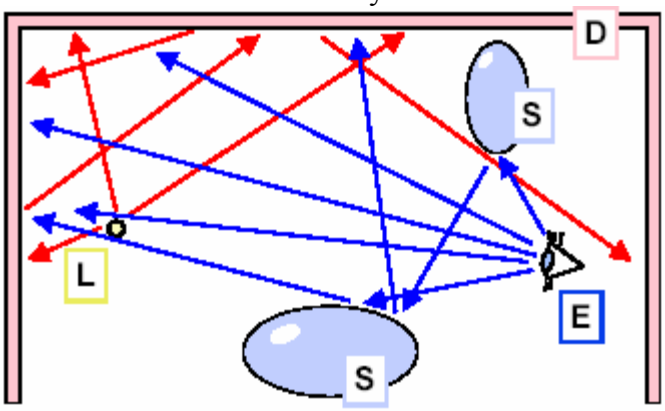
- tento krok není závislý na pozici pozorovatele

- zobrazení sledováním paprsku: $S_M * E$
  - nepočítají se stínovací paprsky => velká rychlost
  - místo světelného modelu se používá **předem spočítaná radiozita**
  - tento krok závisí na úhlu pohledu

Dvoukroková metoda – první krok



Dvoukroková metoda – druhý krok



36.Z jakých omezujících podmínek vychází řešení klasické radiozitní úlohy ?

Zjednodušený fyzikální model Radiozitní rovnice

- přepokládáme **ideálně difusní** povrchy:
  - **BRDF** není závislá na vstupním a výstupním úhlu
  - Výstupní zář **L(x,ω)** nezávisí na směru **ω**

$$L_o(x) = L_e(x) + \int_S f_r(x) L_o(y) G(x,y) V(x,y) dA_y$$

$$L_o(x) = B(x)/\pi \qquad L_e(x) = B_e(x)/\pi \qquad f_r(x) = \rho(x)/\pi$$

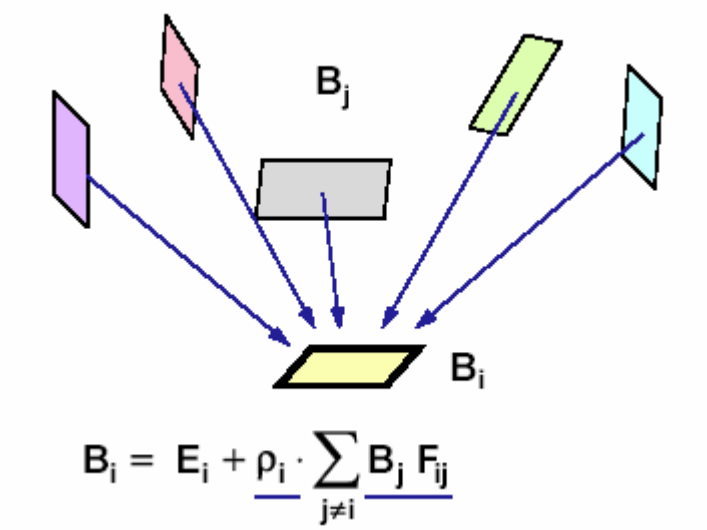
$$B(x) = B_e(x) + \frac{\rho(x)}{\pi} \int_S B(y) G(x,y) V(x,y) dA_y$$

Radiozitní metoda
- vícenásobné difusní odrazy
- nerozlišuje primární a sekundární zářiče
- řešení metodou konečných prvků

- závisí na kvalitě sítě ploch a elementů
- pohledově nezávislé řešení
- světelné cesty: **LD*E**

37. Zformulujte základní soustavu radiozitní úlohy a vysvětlete rozdíl mezi iteračním řešením soustavy pomocí sbírání a střílení energie.

Fyzikální interpretace (sbírání)



Reziduum (odhad chyby) k-té iterace:

$$r^{(k)} = E - M \cdot B^{(k)}$$

V jednom kroku výpočtu se aktualizuje složka vektoru řešení $\mathbf{B_i}$**:**

$$B_i^{(k+1)} = B_i^{(k)} + \frac{r_i^{(k)}}{M_{ii}}$$

(Jacobiho metoda … rezidua se opravují po dokončení iterace, Gauss-Seidel oprava po každém kroku)

Inkrementální výpočet rezidua

Aktualizace vektoru řešení v jednom kroku výpočtu:

$$B^{(p+1)} = B^{(p)} + \Delta B^{(p)}$$

Oprava rezidua:

$$r^{(p+1)} = E - M \cdot \left(B^{(p)} + \Delta B^{(p)}\right) = r^{(p)} - M \cdot \Delta B^{(p)}$$

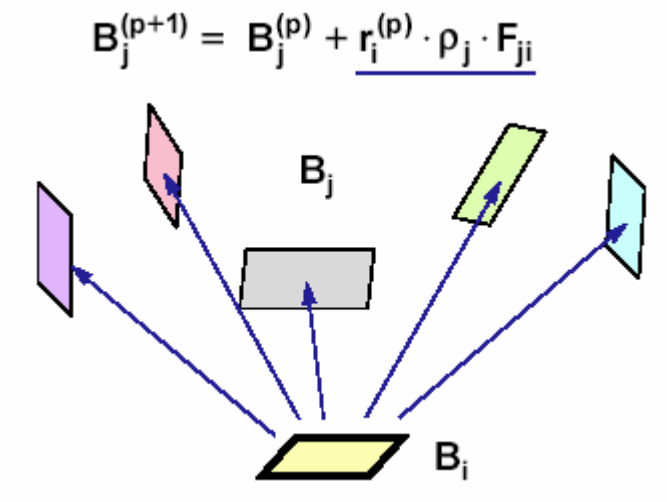
Protože se změnila pouze i-tá složka vektoru řešení:

$$r_j^{(p+1)} = r_j^{(p)} - M_{ji} \cdot \frac{r_i^{(p)}}{M_{ii}} \qquad j = 1..N$$

Fyzikální interpretace (střílení)

- $\mathbf{B_i}$ **…** radiosita i-té plošky (přímá i nepřímá)
- **Jeden krok výpočtu …** rozdělení (výstřel) radiosity z i-té plošky do okolí

- r_i **…** dosud **nevystřelená radiosita** i-té plošky
- **konvergence metody** … celková nevystřelená energie ve scéně se zmenšuje



38. Na jakém principu je založena progresivní radiační metoda?

Progresivní radiační metoda
- M. Cohen a spol.
- Interaktivní výpočet osvětlení
  - Po každém kroku se nakreslí průběžný výsledek
  - Snaha dobře odhadnout řešení již v několika prvních krocích
- Modifikace Southwellovy metody
  - Výběr plošky s největší dosud nevystřelenou energií
  - Použití okolní složky osvětlení

Okolní složka (amgient term)
- vylepšení vzhledu průběžně vykreslených mezivýsledků
- aproximace dosud nespočítaných odrazů světla

Celková dosud nevystřelená radiosita:

$$\overline{\Delta B} = \frac{\sum r_i \cdot A_i}{\sum A_i}$$

$$\bar{\rho} = \frac{\sum \rho_i \cdot A_i}{\sum A_i}$$

Průměrný koeficient odrazu:

Odhad zbytkové (okolní) radiosity:

$$B_{amb} = \overline{\Delta B} \cdot \left(1 + \bar{\rho} + \bar{\rho}^2 + \ldots\right) = \frac{\overline{\Delta B}}{1 - \bar{\rho}}$$

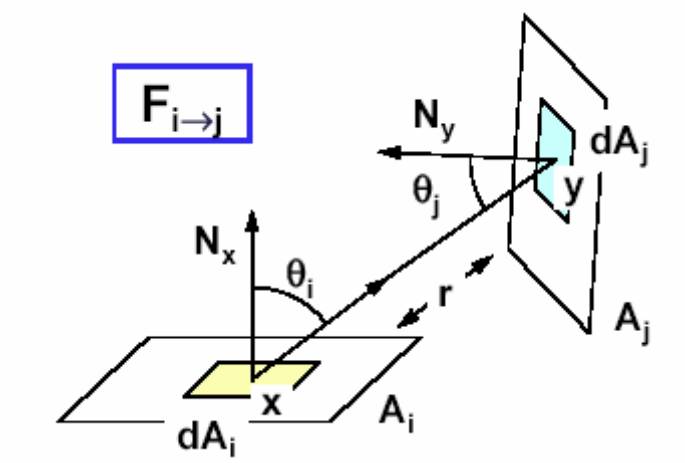
Pro zobrazení se radiosita každé plošky upraví:

$$B_i^{disp} = B_i + \rho_i \cdot B_{amb}$$

39. Geometrická formulace konfiguračních faktorů, praktické metody výpočtu k.f.

Konfigurační faktor $F_{i->j}$

- udává **podíl energie** vyzářené z plochy **i**, která dopadne na plochu **j**
  - klíčová hodnota při sestavování soustavy lineárních rovnic (hledání radiosit jednotlivých ploch)
  - první výpočet (fyzika): Lambert
- závisí pouze na **geometrii scény**
  - vzdálenost, sklon a viditelnost příslušných plošek
- $\mathbf{F_{i->j}}$ je bezrozměrné číslo z intervalu <0, 1>
  - Pro konvexní plošku **i** je $\mathbf{F_{i->i} = 0}$



Rovnice pro radiositu (konstantní elementy):

$$B_i = E_i + \rho_i \cdot \sum_{j=1}^{N} B_j \cdot \frac{1}{A_i} \int_{A_i} \int_{A_j} g(y,x)\, dA_j\, dA_i$$

$$F_{i\to j} = \frac{1}{A_i} \int_{A_i} \int_{A_j} g(y,x)\, dA_j\, dA_i =$$

$$= \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\theta_i \cdot \cos\theta_j}{\pi \|x-y\|^2} \cdot V(x,y)\, dA_j\, dA_i$$

40. Lumigraph - princip metody.

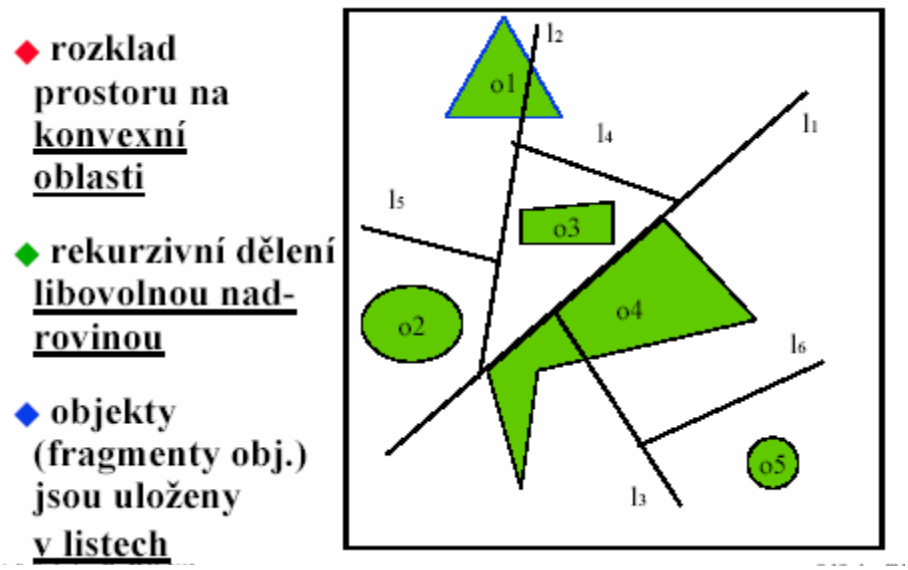
Datové struktury pro prostorové třídění

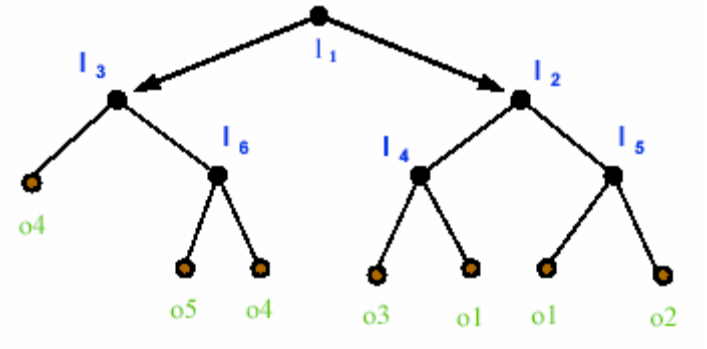
41. BSP stromy, princip, varianty, využití při kreslení malířovým algoritmem.

Binary Space Partition (BSP) trees

- **BSP** stromy jsou binární stromy sloužící k prostorovému třídění scény obecně v **n** rozměrném prostoru

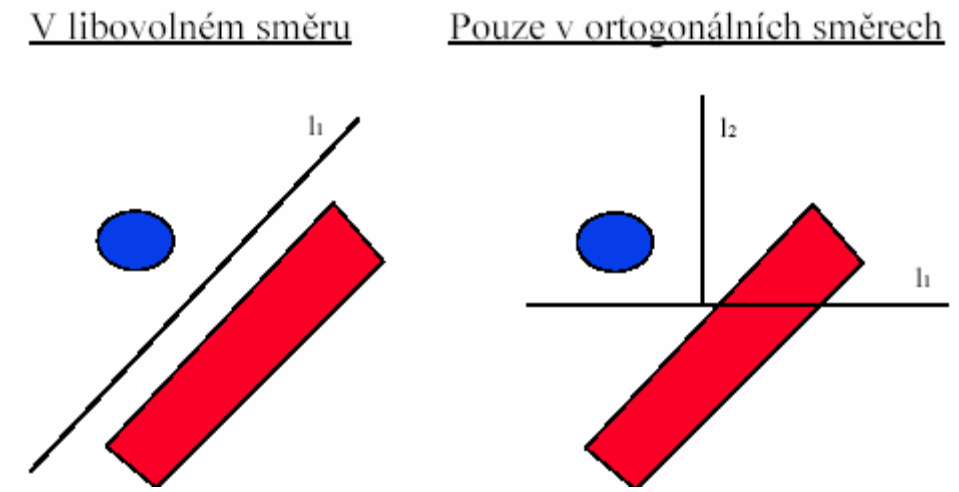
- Celá scéna je rekurzivně dělena nadrovinou
- Listy **BSP** stromu odpovídají konvexnímu rozkladu scény. Prostor určený jedním listem obsahuje maximálně jeden objekt (případně zlomek objektu)



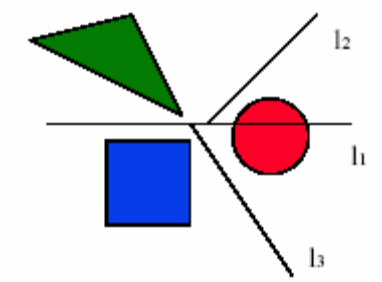


Některé způsoby tvorby BSP stromů

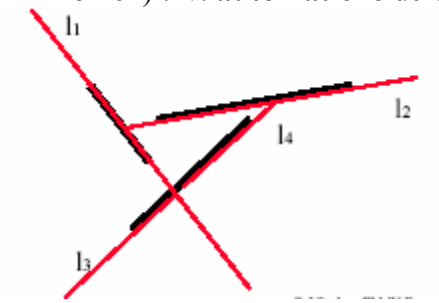
- Dělící nadroviny jsou orientovány:



- jednotlivé objekty mají dimenzi menší nebo rovnu **n** a jsou obsaženy pouze v listech výsledného BSP stromu



- Jednotlivé objekty mají dimenzi **n-1** a každá dělící nadrovina obsahuje některé těleso (jeho zlomek) tzv. **automatické** dělení



Velikosti výsledných BSP stromů

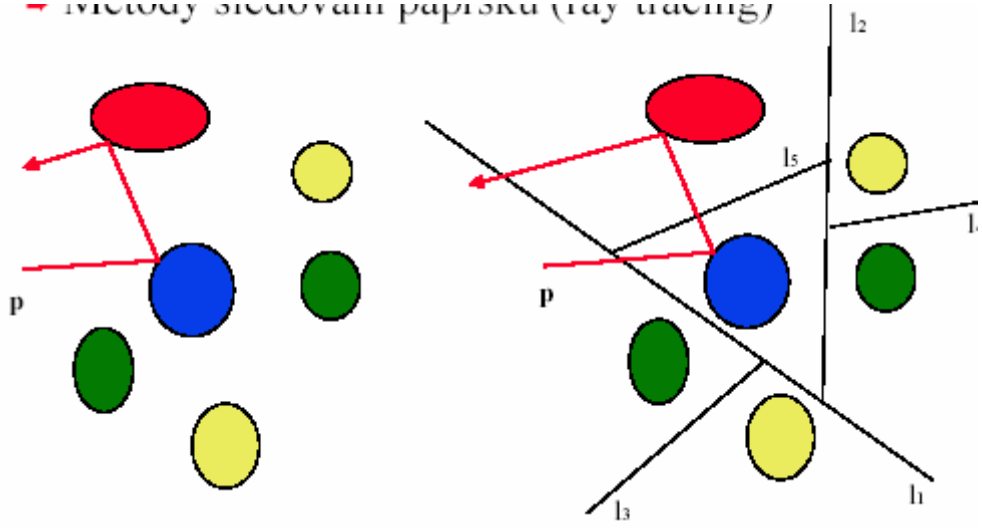
- V rovině: až dok. opt. hranice pro nejhorší případ
  - **ortogonální:** $\Omega(n^2)$ $\Theta(n)$
  - **obecné:** $\Omega(n^2)$ $O(n \log n)$
- V prostoru:
  - **ortogonální:** $\Omega(n^3)$ $\Theta(n\sqrt{n})$
  - **obecné:** $\Omega(n^3)$ $\Theta(n^2)$
- Velké množství náhodně vytvořených BSP stromů se však chová „rozumně“ jejich velikost je podstatně menší nežli tyto stanovené hranice.

42. Zrychlení RT pomocí BSP stromů.

Možnosti využití BSP stromů

```
Algorithm SledujPaprsek(paprsek, BSP);
// Paprsek je aproximován polopřímkou vycházející ze
// vstupního bodu
Begin
  if BSP.list=True
    then Zjisti odrazy paprsku od těles nacházejících se
         v tomto regionu a uprav podle nich jeho dráhu;
         return výsledek;
  else
    begin
      Rozděl  paprsek podle dělící roviny na p1 a p2.
      if Paprsek protíná region uzlu BSP.pravý_podstrom
        then SledujPaprsek(p1, BSP.pravý_podstrom);
      if Paprsek protíná region uzlu BSP.levý_podstrom
        then SledujPaprsek(p2, BSP.levý_podstrom);
    end;
end.
```

- Metody sledování paprsku (ray tracing)



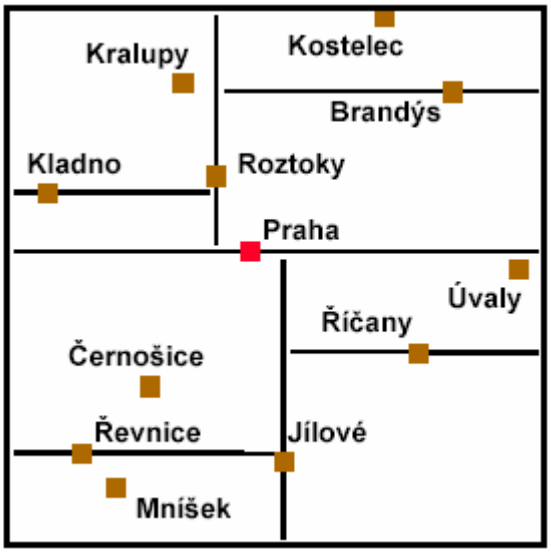
43. Ortogonální rozsahové vyhledávání. Datové struktury a příklady použití.

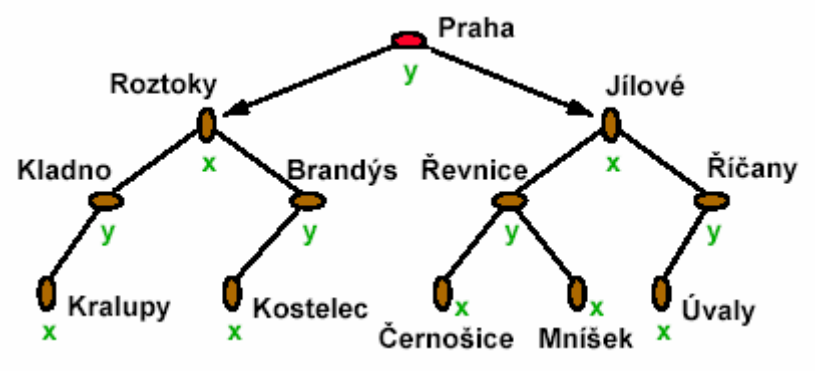
Ortogonální rozsahové vyhledávání

- Ortogonální rozsahové vyhledávání lez řešit pomocí **K-D trees** a **range trees.**
- **K-D tree (K-D strom)**
  - Vyvážený binární vyhledávací strom provádějící dělení střídavě podle x-ové a y-ové souřadnice.
- **Range tree (Rozsahový strom)**
  - Vyvážený binární vyhledávací strom setříděný podle jedné ze souřadnic. Uzel obsahuje buď hledaný bod, nebo podstrom, který je range tree pro nalezený rozsah a následující souřadnici.

K-D strom

- reprezentace bodových objektů
- adaptivní dělení pouze podle jedné složky (binární strom), pravidelné střídání souř. složek
- objekty jsou ve všech uzlech

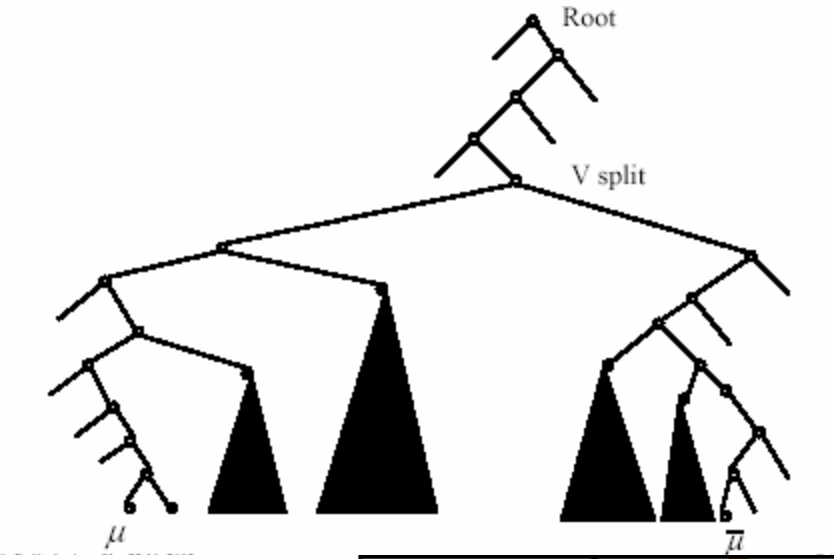




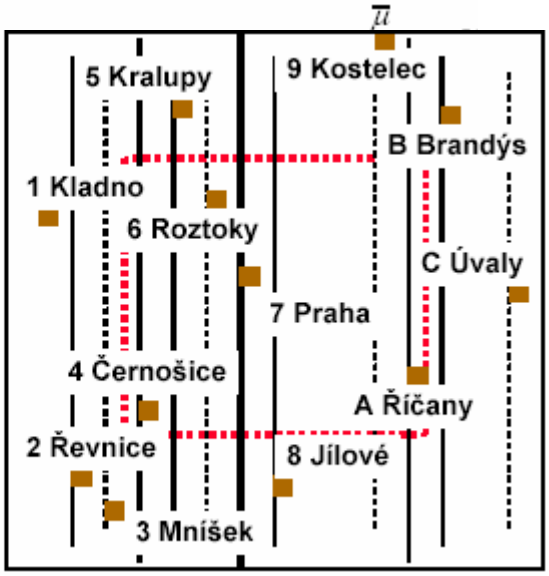
Poznámky o KD-stromech

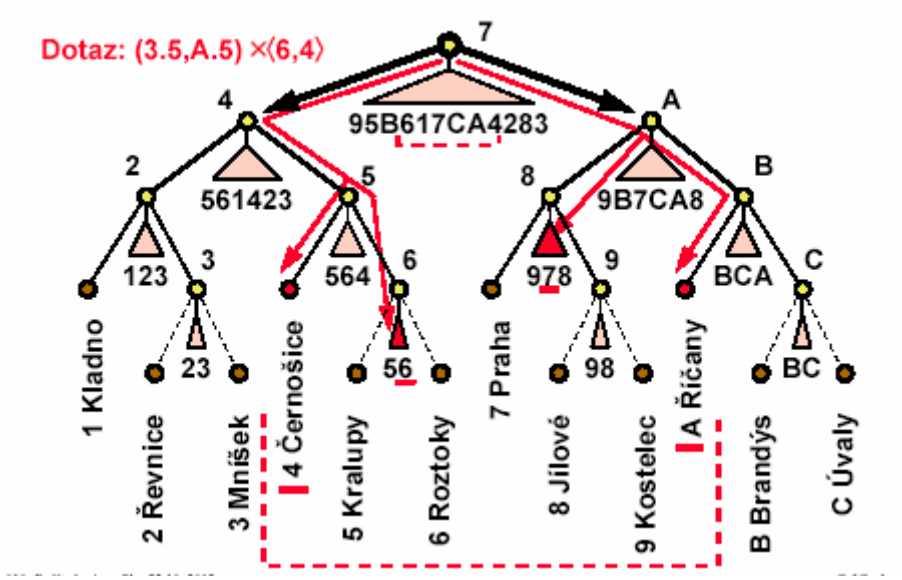
- Region odpovídající uzlu v (region(v)) je obdélník, který může být neomezený v jednom nebo více směrech
- Region kořenu KD-stromu je celá rovina
- Bod je uložen v podstromu uzlu v právě tehdy, jestliže leží v regionu(v).
- Podstrom uzlu v prohledáváme pouze tehdy, jestliže region (v) má neprázdný průnik s dotazovaným obdélníkovým rozsahem.
- Jestliže je region(v) zcela obsažen v dotazovaném rozsahu, může nahlásit všechny body celého podstromu v.

1D Range tree (rozsahový strom)



- vyvážený binární vyhledávací strom podle souřadnice x
- objekty jsou pouze v listech
- vnitřní uzly obsahují 1D rozsahové stromy podle souř. y





Rozsahové vyhledávání: shrnutí

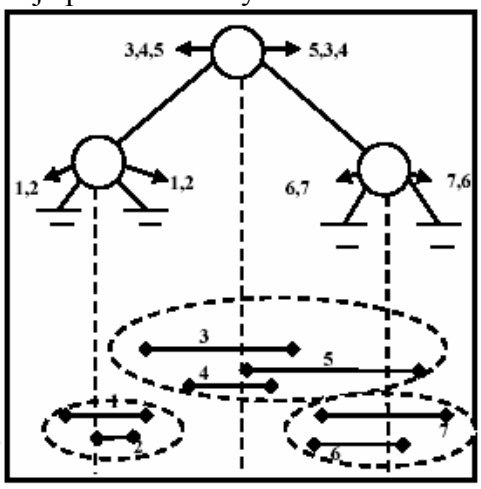
	K-D stromy	Rozsahové stromy
Paměť:	$O(dn)$	$O(n\log^{d-1} n)$
Dotaz:	$O(n^{1-\frac{1}{d}}+k)$	$O(\log^{d} n+k)$
Doba vytvoření:	$O(dn\log n)$	$O(n\log^{d-1} n)$

44. Intervalové stromy, princip a příklad použití.

Intervalové stromy

- Nechť **I** je množina úseček v rovině rovnoběžných s x-ovou osou, **l** je vertikální přímka. **Intervalové stromy** řeší úlohu nalezení všech úseček z množiny **S**, které **l** protíná.
- **Intervalové stromy** pro množinu **I** n intervalů vyžadují O(n) paměti a lze jej vybudovat v čase O(n log n). S použitím Interval tree lze nalézt všechny úsečky protnuté přímkou **l** v čase O(log n + k) kde k je počet nalezených intervalů.

- Vyvážený binární vyhledávací strom
- Objekty jsou uloženy ve všech uzlech a setříděny podle *x*-ové souřadnice.
- Každý uzel obsahuje pouze úsečky protnuté horizont. přím. ve střední hodn. konc. bodů úseček



```
Algorithm ConstructIntervalTree(úsečky);
begin
  if | úsečky | = 0
    then return prázdný list;
    else Vytvoř uzel_v. Vypočti střed množiny koncových bodů
         úseček x a ulož ho do v;
         Vypočti množinu úsečky_střed úseček protnutých
         přímkou s x-ovou souřadnicí x a seznamy bodů
         úsečky_střed_vlevo a úsečky_střed_vpravo setříděné
         podle jejich koncových bodů;
         v.vlevo = ConstructIntervalTree(úsečky_vlevo);
         v.vpravo = ConstructIntervalTree(úsečky_vpravo);
         return v;
end.
```

```
Algorithm QueryIntervalTree(v, qx);
begin
  if v není list
    then Procházej seznamem v.úsečky_střed_vlevo, začínaje
         v nejlevějším bodě směrem doprava a hlas všechny
         úsečky obsahující qx. Skonči jakmile první interval
         neobsahuje qx;
         QueryIntervalTree(v.vlevo, qx);
    else …symetricky pro v.úsečky_střed_vpravo ...
end.
```