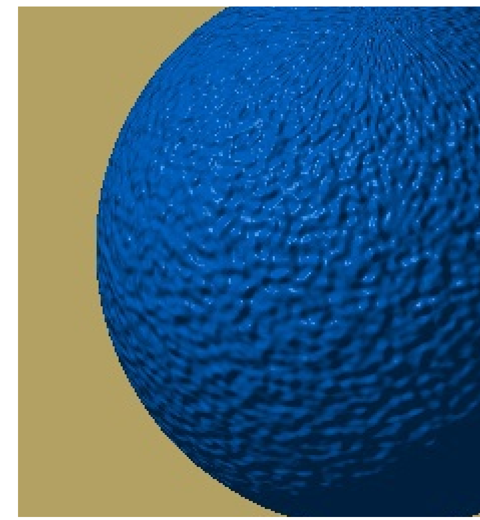
Textury a šumové funkce

© 1998-2016 Josef Pelikán
CGG MFF UK Praha

pepca@cgg.mff.cuni.cz
http://cgg.mff.cuni.cz/~pepca/

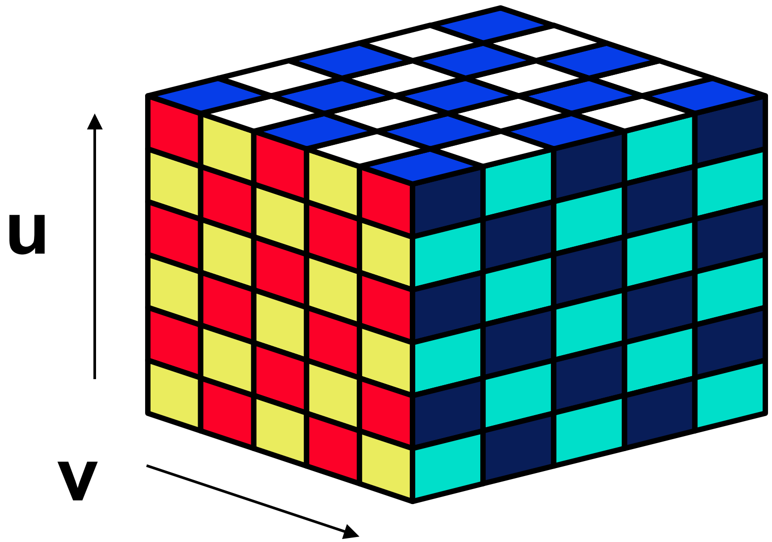
Účinek textury

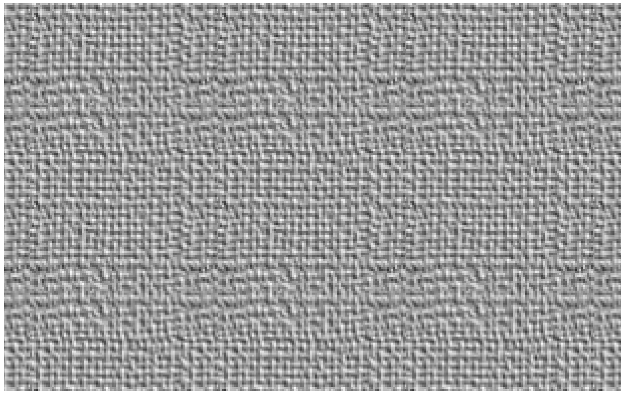
- modulace **barvy** na povrchu předmětů
- změna parametrů **světelného modelu**
  – např. odrazivost ($\mathbf{k}_D$ a $\mathbf{k}_S$), ...
- modifikace **normálového vektoru**
  – „bump-texture”
  – nahrazení složité (mikro)geometrie



- napodobení komplikovaných přírodních jevů
  – náhodné textury (užití syntézy šumu)
  – fraktální textury (deterministické i stochastické)

2D textura





- pokrývá **povrch** tělesa (jako tapeta)
- **mapování textury**: [**x**,**y**,**z**] → [**u**,**v**]
  – inverzní mapovací funkce
- **vlastní textura**: [**u**,**v**] → **barva** (normála, ..)

3D textura

- zachycuje změny veličin **uvnitř tělesa**
- napodobuje **vnitřní strukturu materiálu** (dřevo, mramor, ...)

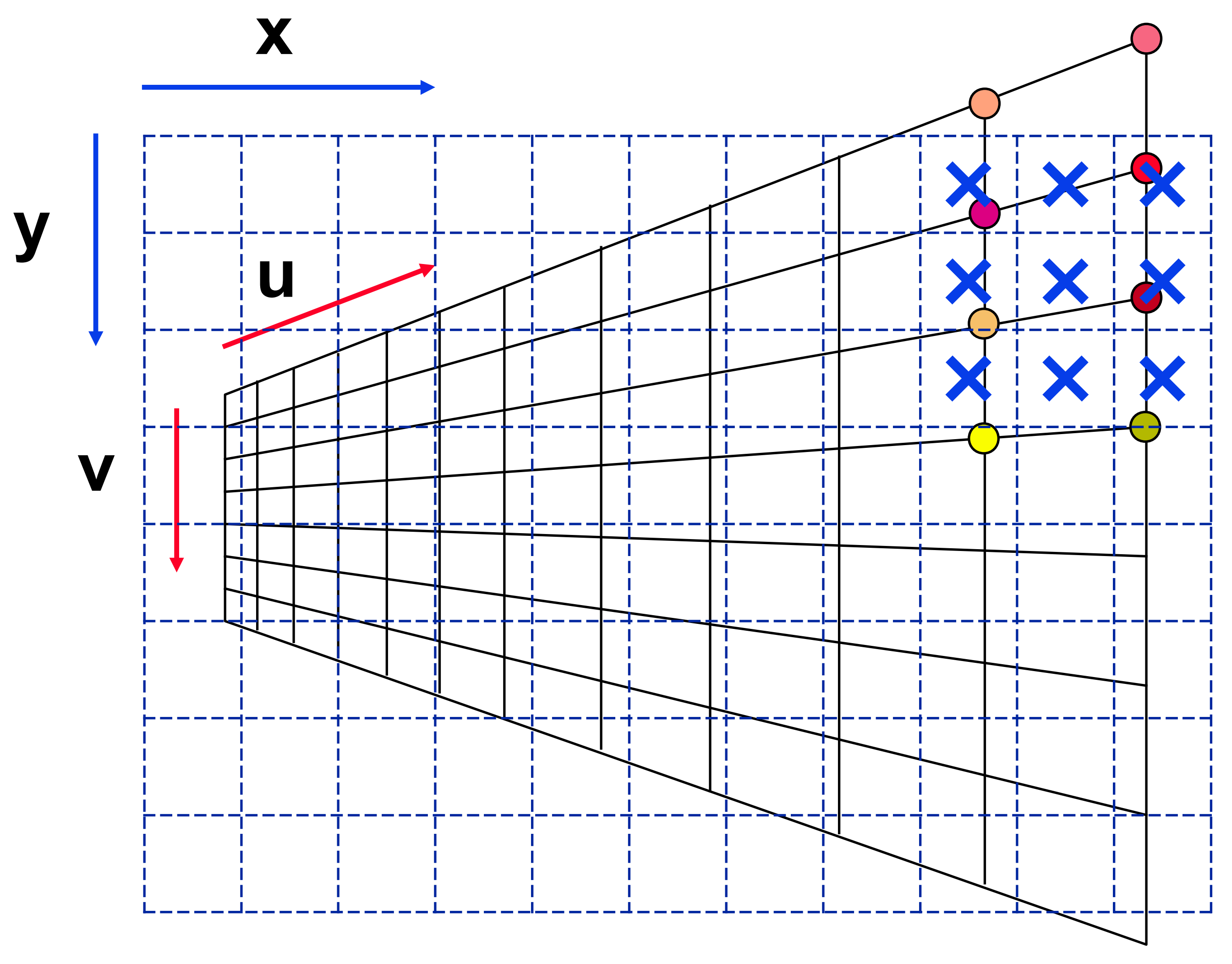
- ➡ není třeba počítat **inverzní mapování**
- ➡ **3D textura**: [**x**,**y**,**z**] → **barva** (odrazivost, atd.)

- pro napodobení přírodních jevů se využívají tzv. **šumové funkce**
  – pseudonáhodné spojité „vrásnění"

Implementace textury

- předem připravené **pole dat** (tabulka, bitmapa)
  - především u 2D textur
  - skutečná (přírodní) data, obrázky, etikety
  - pro větší kvalitu (spojitost) - různé typy interpolace
- textury definované **algoritmem** (předpisem)
  - jednoduché geometrické tvary
  - fraktály, stochastické funkce (šum, turbulence)
- **smíšené metody** (předem spočítaná tabulka)
  - náročné modelovací metody (reakční difuze, ..)

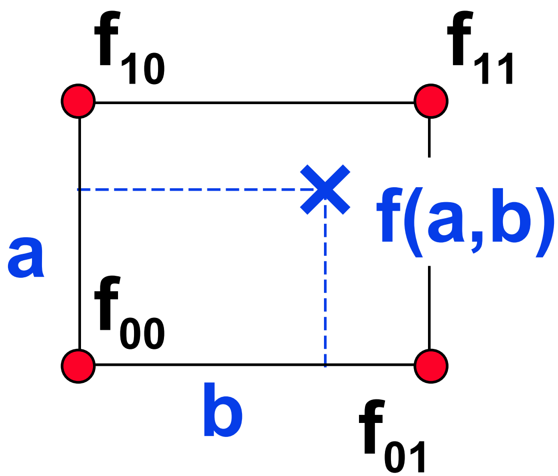
Textura definovaná tabulkou



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)

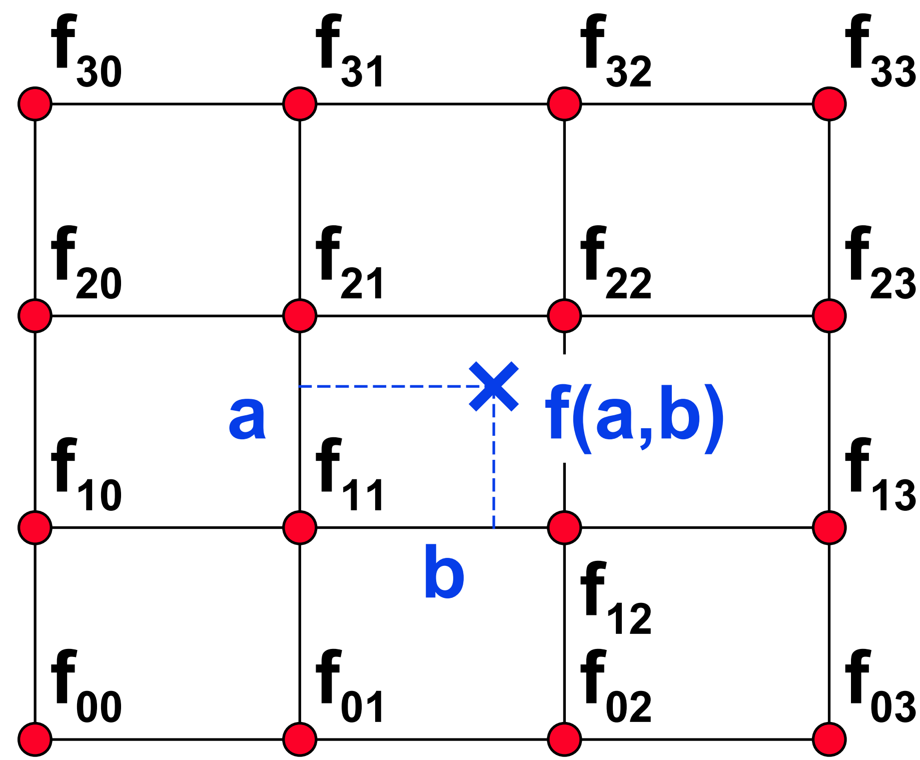
Bilineární a bikubická interpolace



$$f(a,b) = a \cdot \left[b \cdot f_{11} + (1-b) \cdot f_{10} \right] + $$
$$+ (1-a) \cdot \left[b \cdot f_{01} + (1-b) \cdot f_{00} \right]$$

$$f(a,b) = \sum_{i,j=0}^{3} C_i(a)\, C_j(b)\, f_{ij}$$

$C_i(t)$ .. kubické polynomy



Kubická B–spline interpolace

$$f(a,b) = \sum_{i=0}^{3}\sum_{j=0}^{3} C_i(a)\, C_j(b)\, f_{ij}$$

B-spline
váhové funkce:

$$C_0(t) = \tfrac{1}{6}(1-t)^3$$

$$C_1(t) = \tfrac{1}{6}\left(3t^3 - 6t^2 + 4\right)$$

$$C_2(t) = \tfrac{1}{6}\left(-3t^3 + 3t^2 + 3t + 1\right)$$

$$C_3(t) = \tfrac{1}{6}t^3$$

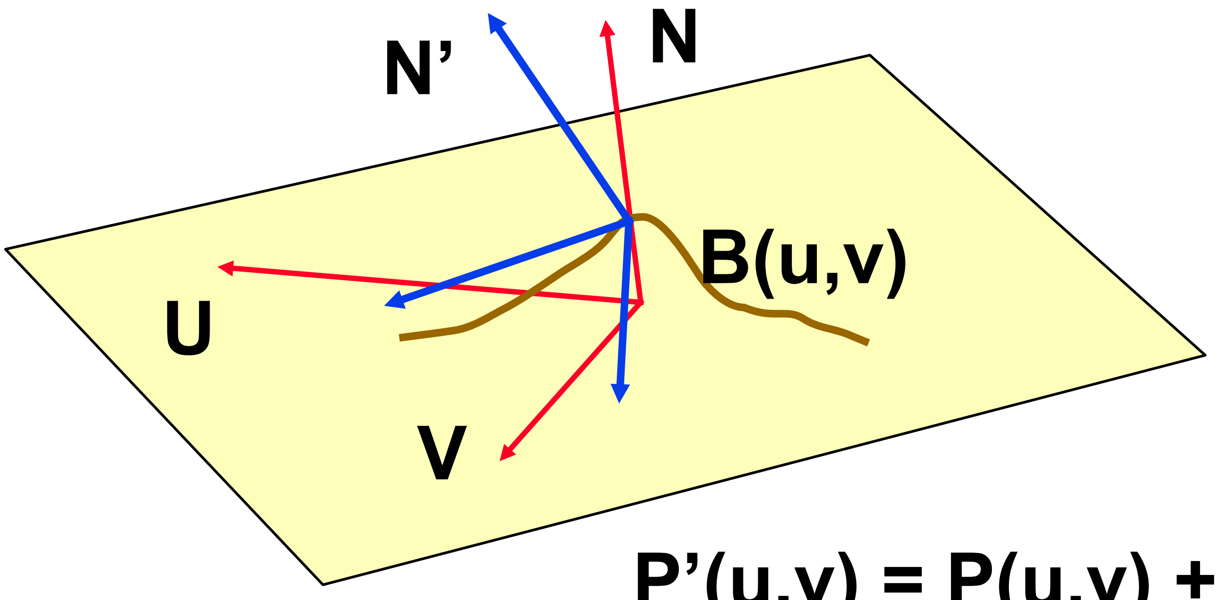
$$\sum_{i=0}^{3} C_i(t) = 1$$

$$0 \le C_i(t) \le 1 \quad \text{pro} \quad 0 \le t \le 1$$

Algoritmické a smíšené textury

- jednoduché **geometrické tvary**, vzorky
  - šachovnice, pravidelné pruhy, ..
- napodobení **přírodních tvarů**
  - často se využívají **pseudonáhodné algoritmy** (syntéza spojité šumové funkce)
  - fraktály, turbulence (mraky, špína, ..)
  - reakční difuze (napodobení kůže a srsti zvířat)
  - prostorové náhodné textury (dřevo, mramor, ..)

Modulace normály („bump map”)

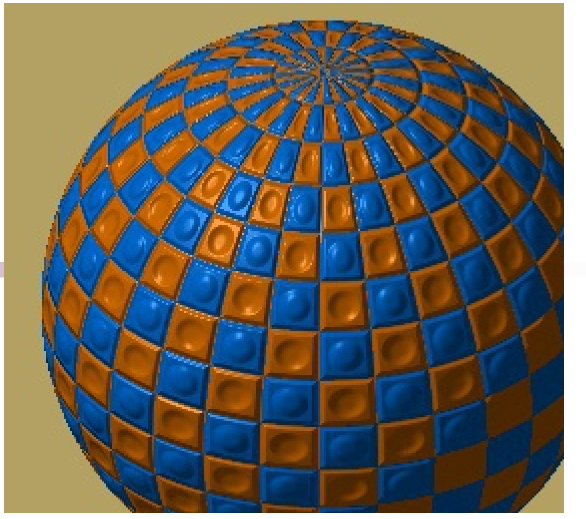


$$\underline{N} = U \times V$$

$$P'(u,v) = P(u,v) + B(u,v) \cdot N / |N|$$

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

Modulace normály



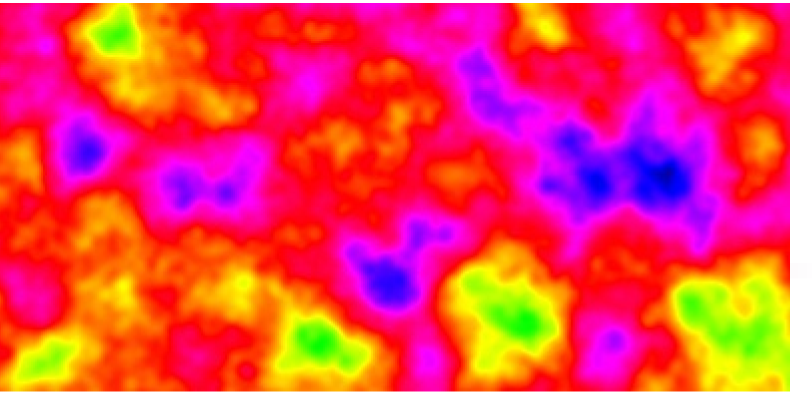
Původní normála: $\mathbf{N} = \mathbf{U} \times \mathbf{V}$

posunutý bod: $\mathbf{P}'(\mathbf{u},\mathbf{v}) = \mathbf{P}(\mathbf{u},\mathbf{v}) + \dfrac{\mathbf{B}(\mathbf{u},\mathbf{v}) \cdot \mathbf{N}}{|\mathbf{N}|}$

aproximace **modifikované normály**:

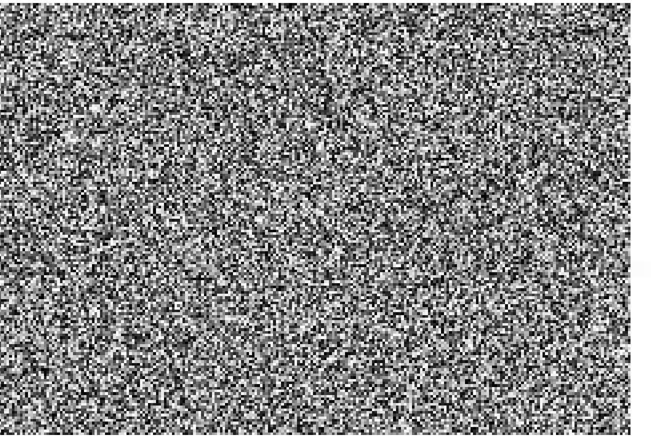
$$\boxed{\mathbf{N}' = \mathbf{N} + \frac{\frac{\partial \mathbf{B}}{\partial \mathbf{u}}(\mathbf{u},\mathbf{v}) \cdot (\mathbf{N} \times \mathbf{V}) - \frac{\partial \mathbf{B}}{\partial \mathbf{v}}(\mathbf{u},\mathbf{v}) \cdot (\mathbf{N} \times \mathbf{U})}{|\mathbf{N}|}}$$

Syntéza šumu



- cílem je **subjektivně nahodilý** vzhled (tvar)
  - napodobení komplikovaných přírodních jevů
  - výsledek působení chaotického systému, náhodné difuse, systému s částečnou zpětnou vazbou, ..
- výpočet hodnoty šumové funkce v konkrétním bodě musí být **deterministický** (opakovatelný)
  - paralelní implementace, opakovaný dotaz
- požadovaná **spektrální charakteristika** šumu
  - nekorelovaný/bílý šum, spojitý šum

Bílý šum



- šum s **neomezeným spektrem**
  – hodnoty v různých bodech mají nulovou korelaci

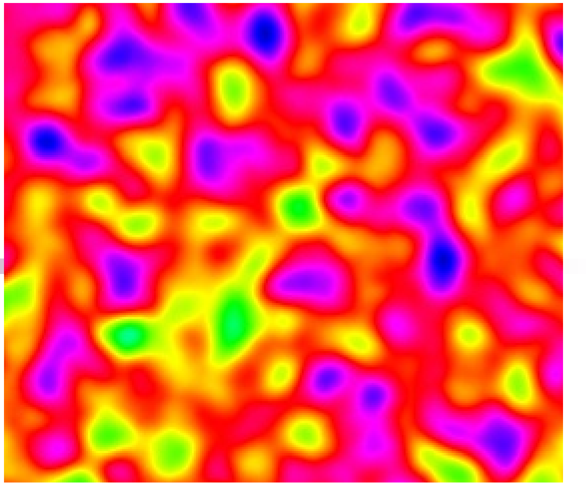
- příklad implementace 2D bílého šumu:

```
double RandomTab[RANDOMTABLEN];        // random values
int Indx[ILEN], Indy[ILEN];            // random permutations

double white_noise_2D ( double x, double y )
{ int i = HASH( Indx[LOWBITS(x)], Indy[LOWBITS(y)] );
  return( RandomTab[ i % RANDOMTABLEN ] );
}
```

využívá **k** bitů mantisy **`LOWBITS`**, hashovací funkci **`HASH`**
`RandomTab`, **`Indx`**, **`Indy`** jsou předem spočítané tabulky

Spojitý šum



- **spojitá funkce** s omezeným spektrem
  – stacionární, izotropní (invarian. na posunutí, otočení)
  – krátká perioda funkce může být na závadu
- **Fourierovská syntéza**
  – přesně mohu ovlivňovat frekvenční charakteristiku
- **interpolace** náhodných hodnot v mřížce
  – klasická interpolace (např. pomocí B-spline funkcí)
  – Hermitovská interpolace - gradienty (Perlin)
  – stochastická mřížka - řídká konvoluce (Lewis)

Interpolace v pravidelné mřížce

- ❶ předem vygeneruji **síť pseudonáhodných čísel** (vektorů, směrnic, ..)
  - požadované rozdělení pravděpodobnosti
  - 1D, 2D nebo 3D topologie
  - ve vícerozměrných případech mohu ušetřit paměť pomocí hašovací funkce **`HASH(x,y,z)`** - viz výše
- ❷ **interpolace** v neuzlových bodech
  - separabilní metody (složky počítám postupně)
  - nejčastěji kvadratické nebo kubické polynomy
  - **2D**: 4 až 16 uzlů, **3D**: 8 až 64 uzlů

Metoda Kena Perlina (3D šum)

- spektrum je omezené na jednu oktávu (**f ÷ 2f**)
- efektivní implementace

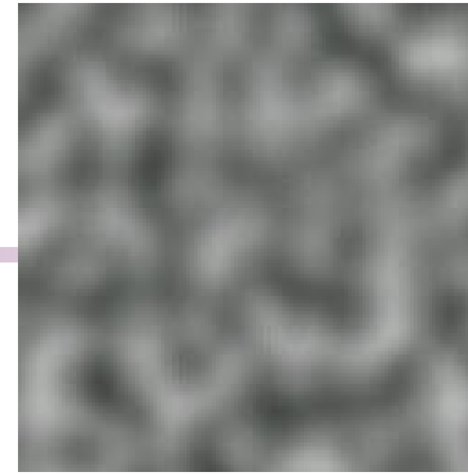
❶ předem vygeneruji **síť pseudonáhodných gradientních vektorů** $\mathbf{[a,b,c,d]_{ijk}}$

– $\mathbf{[a,b,c]_{ijk}}$ je pseudonáhodný **jednotkový směr** (vybírám a normalizuji pouze náhodné vektory kratší než 1)

– $\mathbf{d_{ijk}}$ je hodnota funkce v uzlovém bodě $\mathbf{[x_i,y_j,z_k]}$

– přepočítám $\mathbf{d'_{ijk} = d_{ijk} - a_{ijk} \cdot x_i - b_{ijk} \cdot y_j - c_{ijk} \cdot z_k}$

Metoda Kena Perlina



❷ hodnoty v **uzlech**:

$$K_{ijk}(x,y,z) = d'_{ijk} + a_{ijk} \cdot x + b_{ijk} \cdot y + c_{ijk} \cdot z$$

❸ **interpolační** kubické spline polynomy:

$w(t) = 2|t|^3 - 3t^2 + 1$ pro $|t| < 1$

$w(t) = 0$ jinde

– nosič má poloměr **1** ⇒ potřebuji pouze 2^D uzlů

$$a(x,y,z) = \sum_{i=\lfloor x \rfloor}^{\lfloor x \rfloor + 1} w(x-i) \sum_{j=\lfloor y \rfloor}^{\lfloor y \rfloor + 1} w(y-j) \sum_{k=\lfloor z \rfloor}^{\lfloor z \rfloor + 1} w(z-k) \cdot a_{ijk}$$

Metoda řídké konvoluce (Lewis)

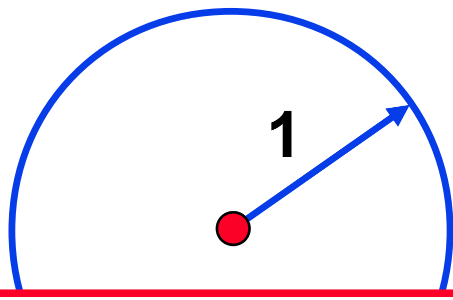
- možnost řízení spektrální charakteristiky
- efektivní implementace

➡ **konvoluce** 3D filtru **h(x,y,z)** s **Poissonovým šumem** γ:

$$n(x,y,z) = \int_{\Re^3} \gamma(u,v,w) \cdot \underline{h(x-u,y-v,z-w)}\, du\, dv\, dw$$

$$\gamma(x,y,z) = \sum_k \underline{a_k} \cdot \delta(x - \underline{x_k}, y - \underline{y_k}, z - \underline{z_k})$$

Konvoluce Poissonova šumu

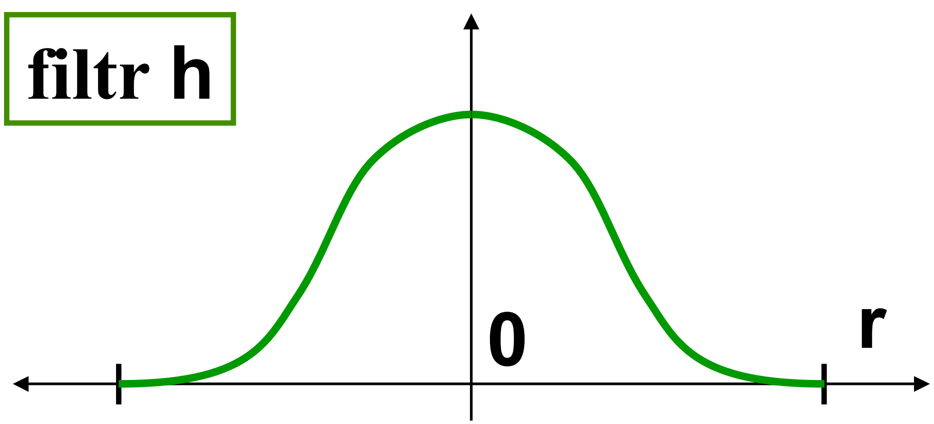


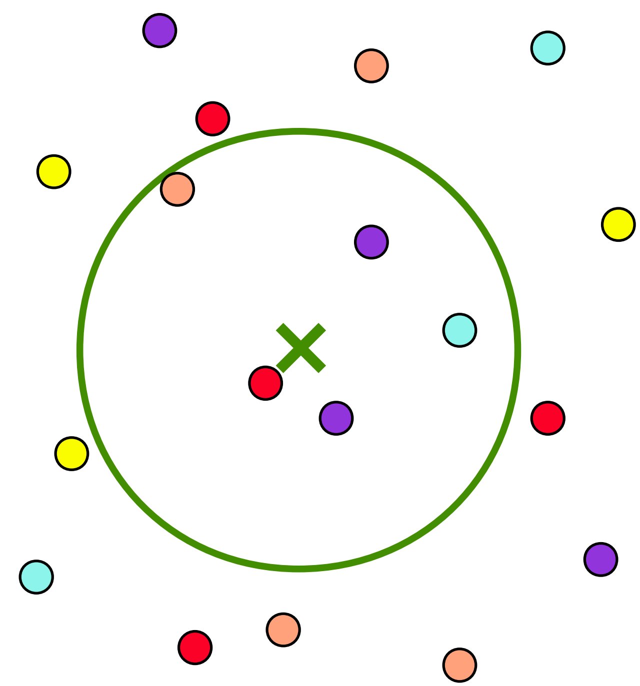
nosič filtru h

náhodná poloha: $[x_k, y_k, z_k]$

náhodná váha: a_k

filtr h





Metoda řídké konvoluce

Díky diskrétnímu charakteru Poissonova šumu:

$$n(x, y, z) = \sum_k \underline{a_k} \cdot h(x - \underline{x_k}, y - \underline{y_k}, z - \underline{z_k})$$

- **hustotou impulsů $[x_k, y_k, z_k]$** mohu řídit kvalitu šumu
  - při **10** a více impulsech na plochu filtru **h** je kvalita téměř nerozeznatelná od interpolačního šumu
  - při stejné kvalitě je řídká konvoluce efektivnější

Efektivní implementace

- prostor rozdělím na krychlové **buňky** se stranou délky **r** (poloměr filtru **h** - obvykle **1**)
- v každé buňce generuji impulsy **nezávisle** pomocí **pseudonáhodného generátoru** s počáteční hodnotou semínka $\mathbf{Seed}_{ijk}$
  - hodnoty $\mathbf{Seed}_{ijk}$ nageneruji předem nějakým jiným (nezávislým) generátorem
  - nebo mohu pro úsporu paměti (a proti opakování vzorku) použít hašovací funkci `HASH(x,y,z)`

Efektivní implementace

- při výpočtu hodnoty **Noise(x,y,z)** stačí projít pouze **několik sousedních buněk**
  - v 2D případě je to **4 ÷ 9** buněk
  - v 3D případě je to **8 ÷ 27** buněk
- pro **izotropní šum** (symetrický filtr **h**) mohu předem spočítat hodnoty **h(r^2)** do tabulky
  - při výpočtu konvoluce nemusím odmocňovat

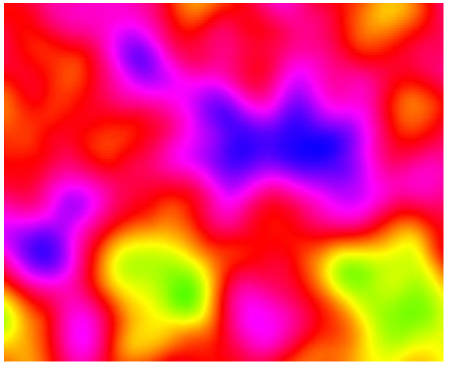
Kombinace šumových funkcí

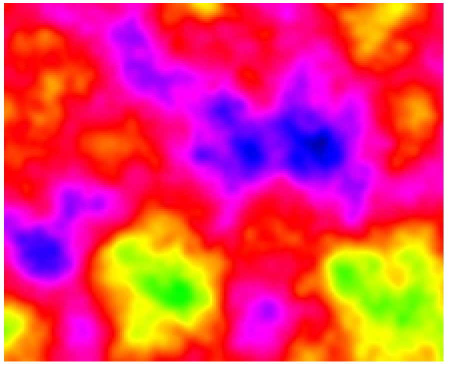
- složení několika frekvencí $\mathbf{f_i}$ s amplitudami $\mathbf{a_i}$ posunutých o vektory $[\mathbf{x_i},\mathbf{y_i},\mathbf{z_i}]$:

$$\sum_i a_i \cdot \mathrm{Noise}\left[f_i \cdot \left(x + x_i \right), f_i \cdot \left(y + y_i \right), f_i \cdot \left(z + z_i \right) \right]$$

- napodobení **turbulence**:

$\mathbf{f_i = F^i}$, $\mathbf{a_i = A^{-i}}$

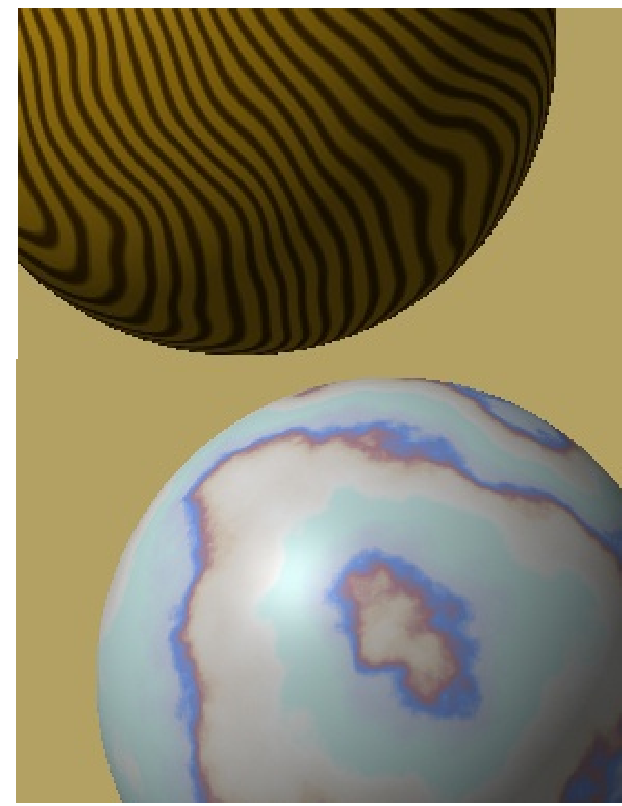




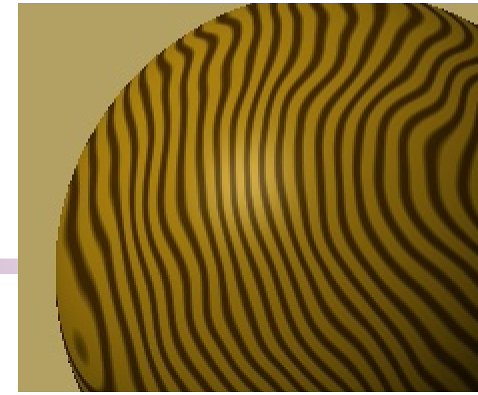
$$\sum_i \frac{1}{A^i} \cdot \mathrm{Noise}\left[F^i \cdot \left(x + x_i \right), F^i \cdot \left(y + y_i \right), F^i \cdot \left(z + z_i \right) \right]$$

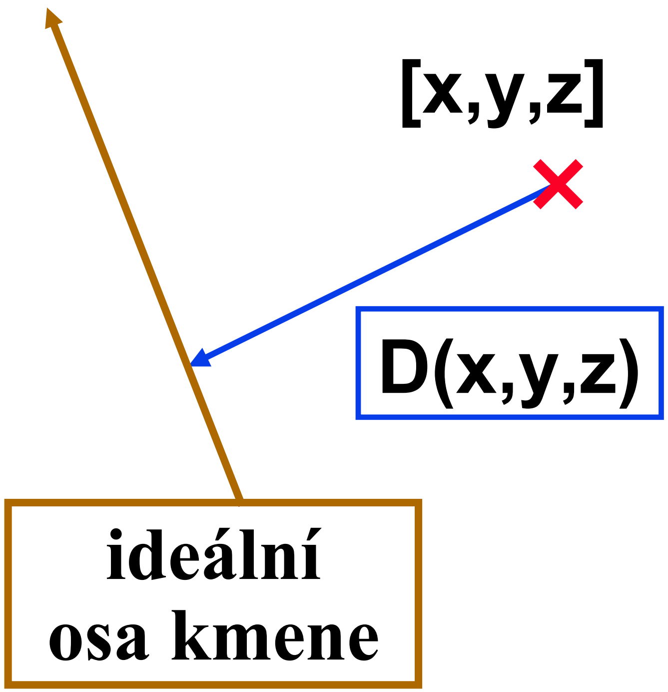
Aplikace šumových funkcí

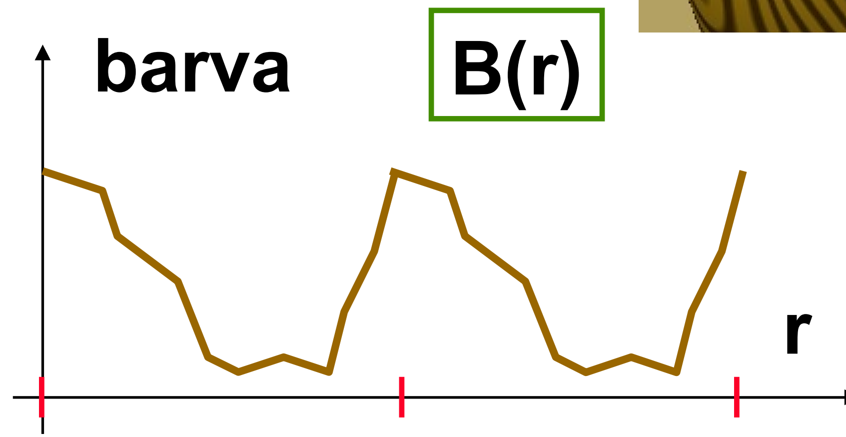
- náhodné **modifikace normály** („bump map”)
  - – iluze nepravidelně zvrásněného povrchu těles
  - – „pomerančová kůra”
- **turbulence**
  - – mlha, mraky, použití v modelování
- **3D textury**
  - – vnitřní struktura materiálu
  - – dřevo, mramor, ..



Simulace struktury dřeva



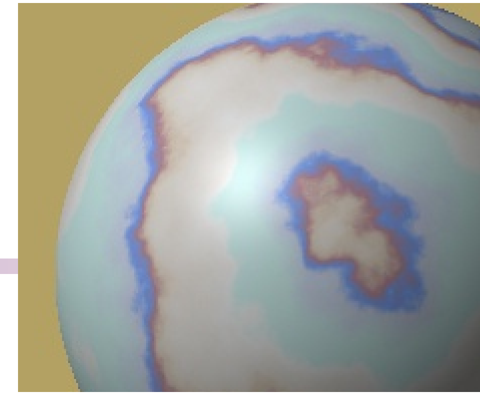


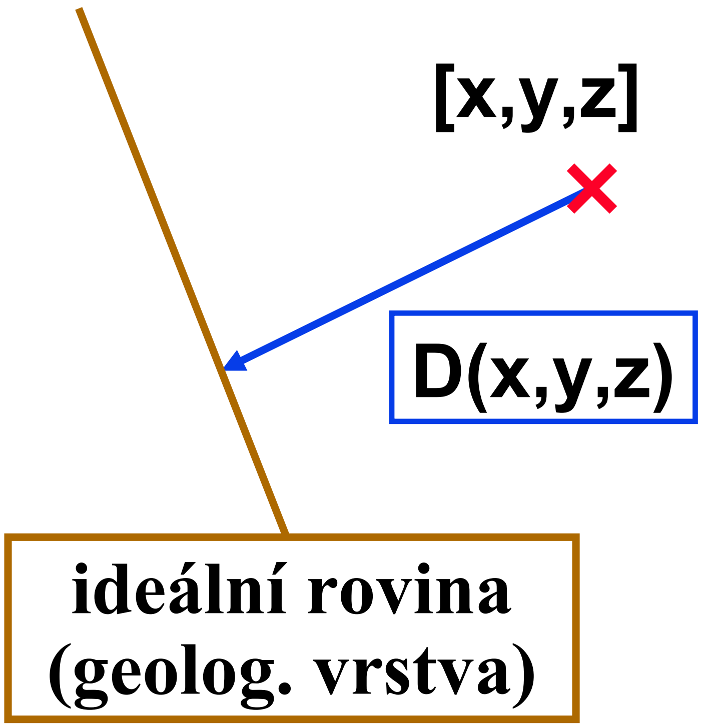


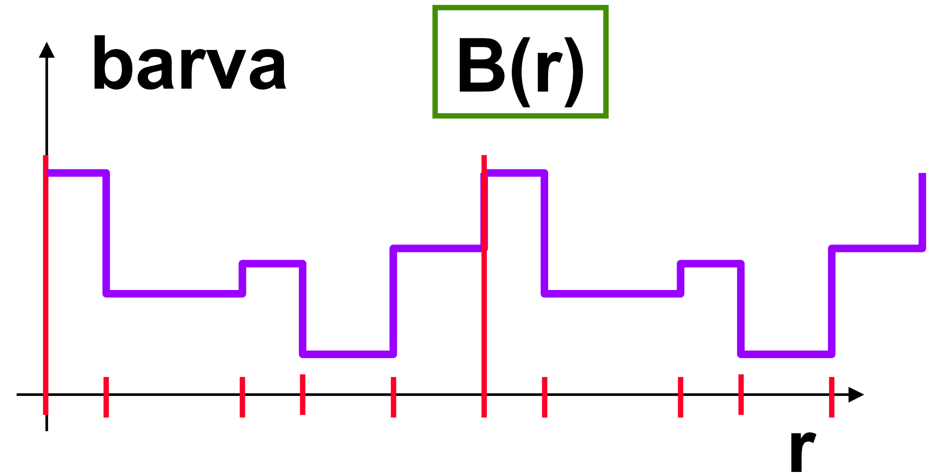
$$B\left[D\left(x, y, z \right) + \mathrm{Noise}\left(x, y, z \right) \right]$$

$$B\left[D\left(x, y, z \right) \cdot \left(1 + \mathrm{Noise}_1\left(x, y, z \right) \right) + \mathrm{Noise}_2\left(x, y, z \right) \right]$$

Simulace struktury mramoru







$$\underline{B\left[D\left(x, y, z \right) + \mathrm{Turb}\left(x, y, z \right) \right]}$$

Literatura

- **K. Perlin**: ***An Image Synthesizer***, Computer Graphics, Vol. 19, #3, July 1985, 287-296
- **K. Perlin, E. M. Hoffert**: ***Hypertexture***, Computer Graphics, Vol. 23, #3, July 1989, 253-262
- **J. P. Lewis: *Algorithms for Solid Noise Synthesis***, Computer Graphics, Vol. 23, #3, July 1989, 263-270
- **J. Foley, A. van Dam, S. Feiner, J. Hughes: *Computer Graphics, Principles and Practice***, 741-745, 1015-1018, 1043-1047