Realtime zobrazování vodní hladiny na dnešních GPU

Jan Horáček

Obsah

- Simulace přírodních efektů
- „Statické“ techniky
- „Dynamické“ techniky
- Implementace
- Otázky a ukázky demoprogramů

Simulace přírodních efektů

Simulace přírodních efektů

- Snaha o co nejrealističtější zobrazení přírody okolo nás (té části, s kterou se denně setkáváme)
- Předmět zájmu:
  - Tráva a jiné rostliny
  - Voda (vodní hladina)
  - Obloha (mraky, rozptylování světla v atmosféře, mlžný opar)
  - Srst zvířat

Simulace přírodních efektů

- Předmětem simulace jsou většinou aktivní (měnící se) děje
- Dají se nějak matematicky popsat a zjednodušit, přičemž výsledek vypadá stále věrohodně
- Nechceme co „nejpřesnější“ simulaci, cílem je co „nejhezčí“ výsledek (z estetického hlediska)

Simulace přírodních efektů

- Některé děje se již delší dobu celkem věrohodně dají simulovat i na starších grafických kartách
  - Mlha
  - „Statické“ mraky v dálce
  - Jednotlivé rostliny s nepříliš složitou strukturou
- Jiné se dají zobrazovat na nejnovějším HW
  - Vodní hladina odrážející i lámající světlo
  - Dynamické mraky, kterými se dá realisticky prolétávat (jen některé typy)

Simulace přírodních efektů

- Ale stále existuje spousta problémů neřešitelných v reálném čase s uspokojivým výsledkem
- Dají se počítat „offline“, např. v animovaných filmech
  - Tráva
  - Srst zvířat
  - Korektní lom světla u průhledých materiálů

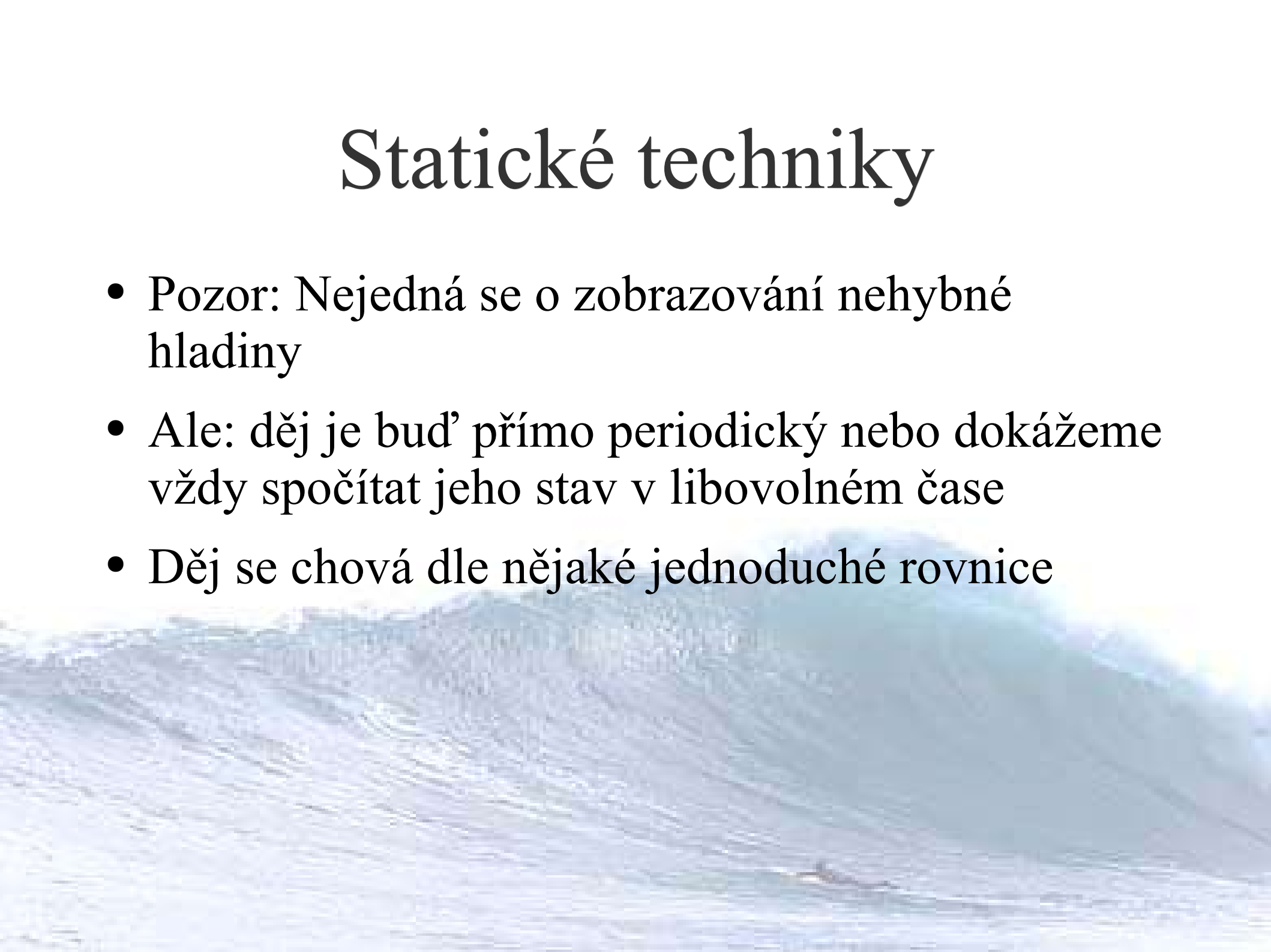
Využití simulace

- Počítačové hry
- Virtuální realita
- ???
- Realtimová esteticky dobře vypadající simulace je dnes vhodná pouze pro vytváření virtuálních světů, ve kterých se člověk cítí „jako doma“
- Obdobně jako filmy - jde hlavně o komerční využití

Statické techniky zobrazování vodní hladiny

Statické techniky

- Pozor: Nejedná se o zobrazování nehybné hladiny
- Ale: děj je buď přímo periodický nebo dokážeme vždy spočítat jeho stav v libovolném čase
- Děj se chová dle nějaké jednoduché rovnice



Výhody

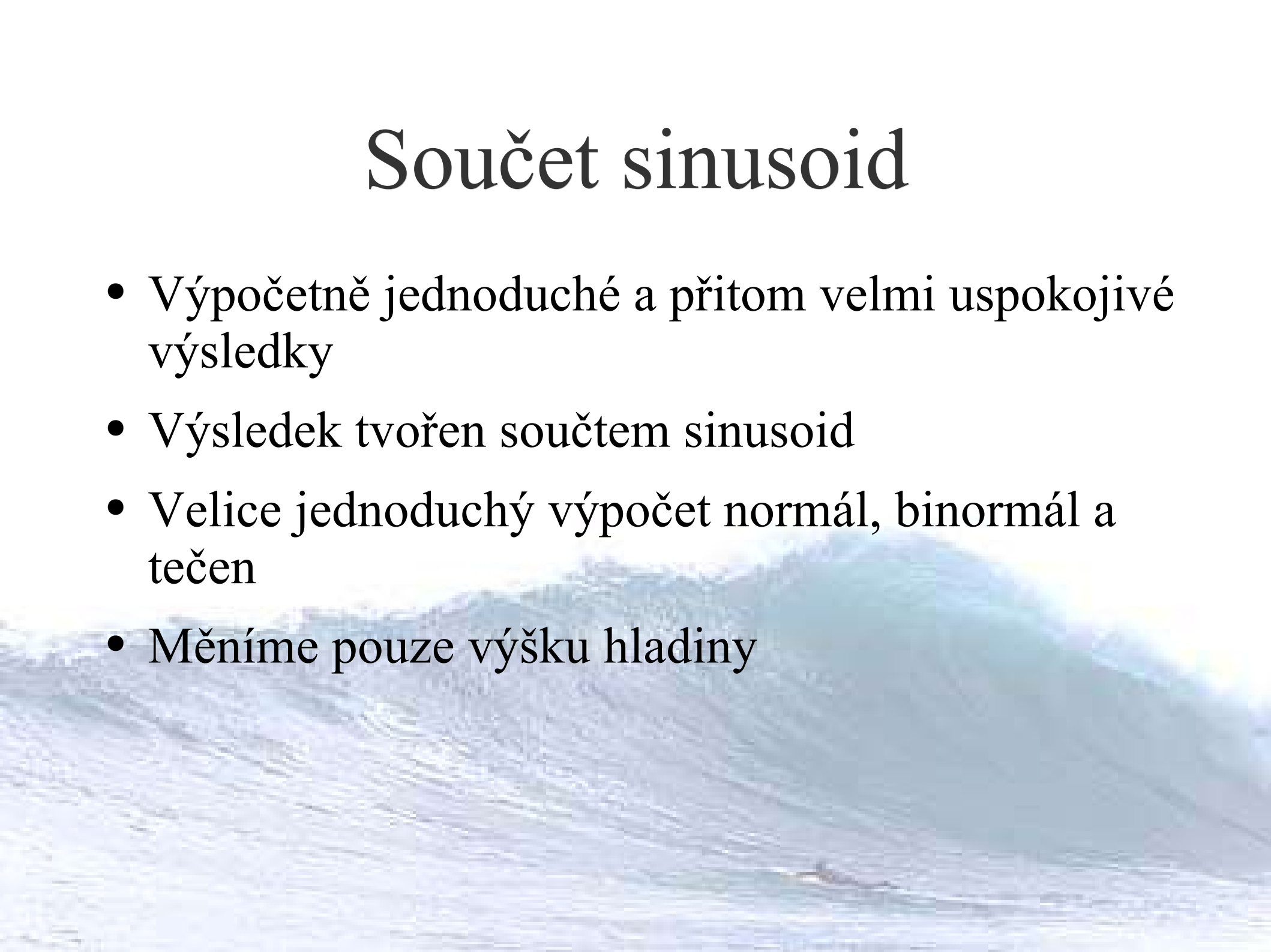
- Často jednoduchý výpočet
- Pro získání normál provedeme analytický výpočet
- Normály, binormály a tečny jsou přesné
- Můžeme celkem snadno kontrolovat úroveň detailů
- Stabilita

Nevýhody

- Špatně reaguje na změny v prostředí
- Není interaktivní
- Špatně se vytváří lokální změny (například kolem pobřeží)

Součet sinusoid

- Výpočetně jednoduché a přitom velmi uspokojivé výsledky
- Výsledek tvořen součtem sinusoid
- Velice jednoduchý výpočet normál, binormál a tečen
- Měníme pouze výšku hladiny



Vzorec

- Vzorec pro výpočet výšky:

$$W(x,y,t) = A * \sin(D \bullet (x,y) * w + t * \varphi)$$

W … výsledná výška vlny na pozici x,y v čase t

A … amplituda vlny

D … směr šíření vlny po povrchu

w … frekvence, $w = 2\pi / L$, kde L je délka vlny

φ … fázová konstanta, $\varphi = S * 2\pi / L$, S je rychlost pohybu vlny ve směru D

Vzorec

- Jedna vlna nám velmi efektivní výsledek nedá, ovšem kombinace několika již dává o mnoho lepší dojem
- Vzorec:

$$H(x,y,t) = \Sigma W_i(x,y,t)$$

- Protože výsledná funkce je suma funkcí, derivace je též suma derivací jednotlivých funkcí

Derivace a tečny

- Parciální derivace podle x (obdobně y):

$$d/dx(W(x,y,t)) =$$

$$w * D.x * A * \cos(D\bullet(x,y) * w + t * \varphi)$$

- Binormála:

$$B(x,y) = (1, 0, d/dx(H(x,y,t))$$

- Tečna:

$$T(x,y) = (0, 1, d/dy(H(x,y,t))$$

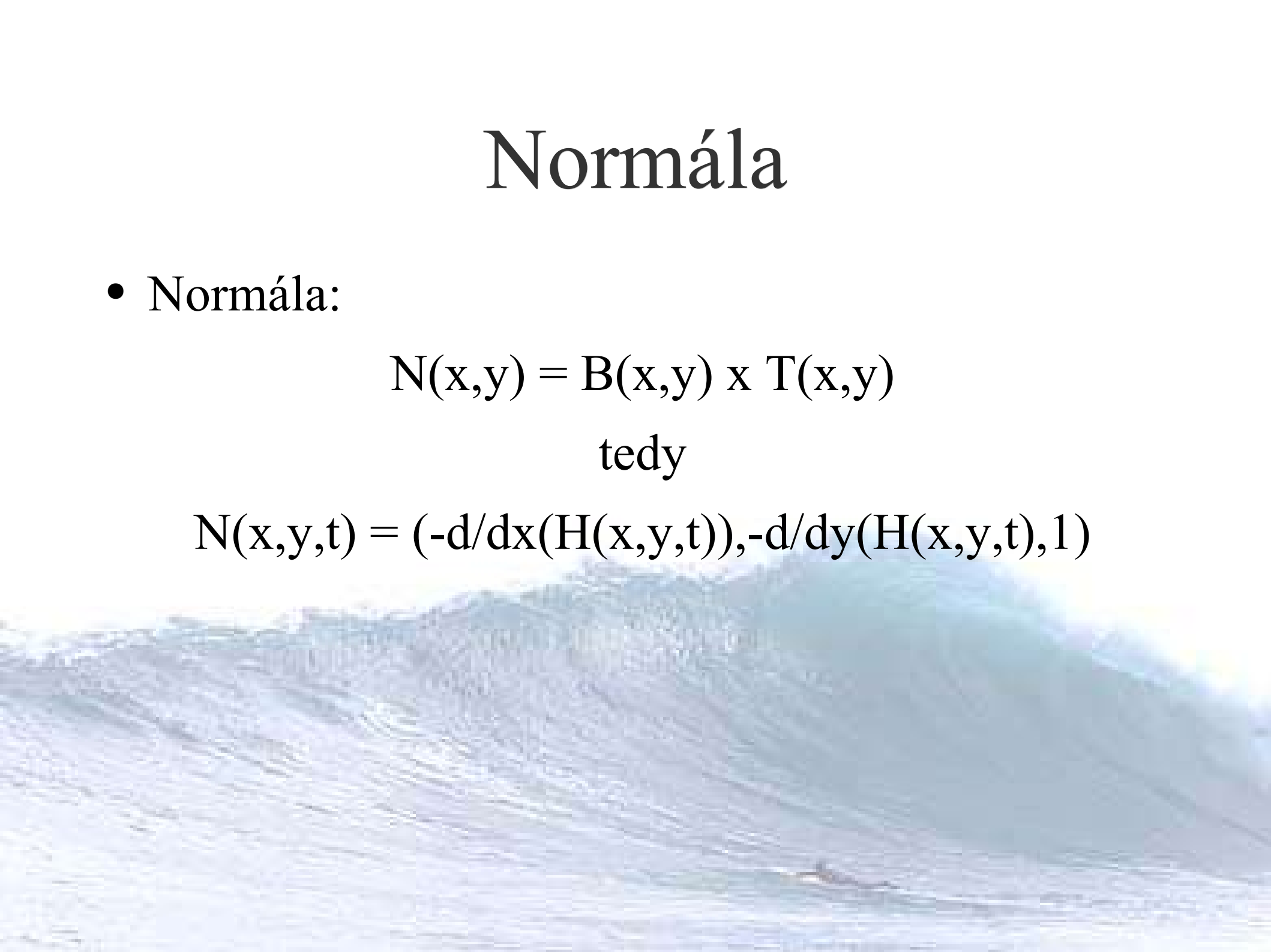
Normála

- Normála:

$$N(x,y) = B(x,y) \times T(x,y)$$

tedy

$$N(x,y,t) = (-d/dx(H(x,y,t)),-d/dy(H(x,y,t),1)$$



Výhody a nevýhody

- Výpočet se dá zjednodušit několika mezivýpočty, takže počítání tečných vektorů a normály je velice nenáročné
- Tvar je dobrý pro simulování například klidného jezírka
- Reálné mořské vlny ovšem mají ostřejší vrcholky a širší „údolí“, to se dá zlepšit například vyšší mocninou funkce sinus:

$$f(x) = 2((\sin(x) + 1) / 2)^k$$

Gerstnerovy vlny

- Lepší tvar pro modelování mořských vln
- Dá se dobře kontrolovat „ostrost“
- Vrcholek vlny je aproximován více vrcholy, dno vlny je širší a je aproximováno méně vrcholy
- Vymyšleny dávno před dobou dnešních počítačů (F.J.Gerstner, 1802) pro simulaci vln na fyzikálním základu

Vzorec

- Pro naše účely budeme vycházet ze vzorce pro sinusové vlny:

$$P(x,y,t) = ($$

$$x + \Sigma(Q_iA_i * D_i.x * \cos(D\bullet(x,y) * w_i + t * \varphi_i)),$$

$$y + \Sigma(Q_iA_i * D_i.y * \cos(D\bullet(x,y) * w_i + t * \varphi_i)),$$

$$\Sigma(A_i * \sin(D_i\bullet(x,y) * w_i + t * \varphi_i)))$$

Normála

- Uvádím pouze vzorec pro normálu, tečnu i binormálu si jistě každý sám dokáže dopočítat

$$N(x, y, t) = ($$

$$- \Sigma(D_i.x * w_i * A_i * \cos(D \bullet P * w_i + t * \varphi_i)),$$

$$- \Sigma(D_i.y * w_i * A_i * \cos(D \bullet P * w_i + t * \varphi_i)),$$

$$1 - \Sigma(Q_i * w_i * A_i * \sin(D_i \bullet P * w_i + t * \varphi_i)))$$

Vlastnosti

- Ostrost vrcholků je dána posunem vrcholů aproximující sítě směrem k vrcholku
- Koeficient Q udává strmost vln
- Pozor: strmost nesmí přesáhnout určitou mez, poté se začnou na vrcholku tvořit chyby
- Opět máme analyticky přesné normály, tečny i binormály

Dynamické techniky zobrazování vodní hladiny

Dynamické metody

- Hlavní požadavek: interaktivita
- Dokáží automaticky reagovat na změny ve virtuálním světě
- Výpočetně složitější pro větší vodní plochy
- Vhodné například pro lokální dopočítávání dohromady s nějakou statickou metodou

Buněčný automat

- Vycházíme ze „hry života“, čtvercová síť buněk, každá buňka je jeden bod na povrchu vody
- Buňky se navzájem ovlivňují
- Musí být nějaký počáteční impuls, jinak je hladina úplně klidná
- Ve chvíli, kdy známe stav systému v čase t, spočítáme stav v čase t+1
- Můžeme dodávat další impulsy a ovlivňovat tak výsledek

Interaktivita

- Část sítě může mít jiné vlastnosti a působit tak jako například nějaká překážka či břeh
- V případě překážek celkem pěkně simuluje i Huygensův princip šíření vln

Algoritmus

- Každá buňka nese informace:
  - Rychlost V
  - Výška H
  - Síla F (počítaná z výšek sousedů)
- Mezi buňkami působí „pružiny“ o tuhosti k
- Pružiny táhnou buňky mezi sebou tak, aby se vyrovnaly jejich výšky

Vzorec

- Přecházíme ze známého stavu (V,H) v čase t do nového stavu (V',H')
  - $F = k * (H1 + H2 + H3 + H4 – 4 * H0)$
  - $V' = V + c1 * F$
  - $H0' = H0 + c2 * V'$
- c1 a c2 jsou koeficienty závislé na čase a hmotnosti

	H1	
H2	H0	H4
	H3	

Stabilita

- Prakticky se využívá pixel shaderů pro výpočet, což kvůli výkonu znamená diskretizaci na 8bitů
- Systém se může stát nestabilní
- Pro lokální výpočet stačí tlumený systém
- Pro globální systém se používá kombinace tlumený systém + dodávání energie z vnějšku
  - Není stabilní, ale chování se dlouhodobě nemění
  - Jednodušší a rychlejší než metody zachovávající stabilitu

Implementace

Implementace

- Efekt se skládá ze dvou částí:
  - Animace vrcholů
  - Animace na úrovni fragmentů
- Vertexové shadery dnes zvládají programy o několik řádů složitější než fragmentové shadery
- Pro animaci vertexů tedy můžeme využít mnohem složitější funkce
- Buněčný automat se naopak dobře implementuje do fragmentového shaderu

Implementace

- Vertex shader animuje přímo vrcholy, potřebujeme tedy metodu, která dává kvalitní tvar na horizontu – např. Gerstnerovy vlny
- Fragment shader už nemění samotnou geometrii, nejsou tolik znát jemné detaily, ale potřebujeme aktivní nenáročnou metodu, která je stále v pohybu

Fragmentový shader

- Výsledkem shaderu je:
  - výšková mapa (pro displacement mapping)
  - normálová mapa (pro bump/environment mapping)
- Globální vlnění: sinusové vlny nebo buněčný automat (doplněný o dodávání energie)
- Buněčný automat pro dopočítání důležitých míst, například vlnění vytvořené pohybem uživatele

Fragmentový shader

- Práce probíhá na textuře nižšího rozlišení, např. 256x256
- Pro globální vlnění se textura opakuje
- Pro lokální dopočítávání se přičítá další textura přes výsledek z ostatních
- Může se použít i více průchodů

Výsledek

- Normálová mapa se dá využít několika způsoby:
  - Bumpmapping
  - Environment mapping
  - Odraz světla od hladiny (dynamický environment mapping)
  - Lom světla (obdoba environment mappingu)
  - Disperze světla

Otázky

Zdroje

- http://www.ati.com/developer/
- http://developer.nvidia.com/
- Randima Fernando: *GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics*, Addison-Wesley, 2004
- Wolfgang F.Engel: *Direct3D ShaderX: Vertex and Pixel Shader Tips and Tricks*,Wordware Publishing, 2002