



Realtime stíny v OpenGL

Tomáš Skácel, 2003



Motivace

- stínování
 - pomáhá porozumět tvaru objektu
- stíny
 - pochopení hloubky scény a prostorových vztahů mezi objekty
 - realističnost scény
 - další z prostředků pro zvýšení atmosféry



Motivace

- donedávna mnoho her dynamické stíny nepoužívalo
 - chyběl robustní algoritmus
 - často různé triky, bez self-shadowing
- časy se mění (Blade of Darkness, pozdější verze Quake3 engine, Doom III)
 - proč se nepřidat



Různé metody

- Light Maps
 - pouze statické osvětlení, ray-casting, radiosity
- Projected Planar Shadows
 - dynamické, ale bez self-shadowing
- robustní algoritmy
 - Shadow Volumes (1977, Frank Crow)
 - Shadow Mapping (1978, Lance Williams)

Shadow Volumes



Shadow Volumes

- ideální bodové nebo směrové zdroje světla
- metoda produkuje ostré stíny
- *self-shadowing*
- víceprůchodový algoritmus

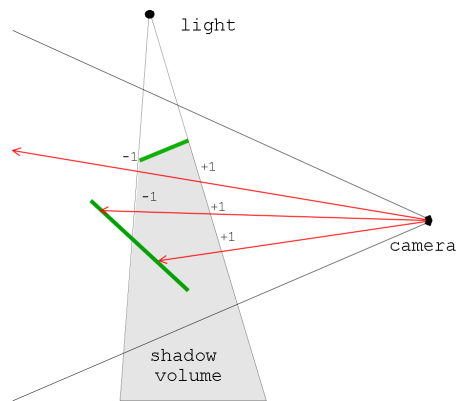
Shadow Volumes

- Princip metody
 - vytvoření pomocného stínového tělesa (*shadow volume*), které je určeno světelným zdrojem a tělesem, které vrhá stín
 - pro další polygony se testuje, zda se nachází uvnitř stínového tělesa

Shadow Volumes

- test, zda polygon leží ve stínu
 - ve *window space*
 - pro každý pixel průmětny se spočítá, kolikrát paprsek během cesty od pozorovatele k prvnímu viditelnému fragmentu vstoupil a kolikrát opustil stínové těleso
 - pokud je počet vstupů větší než počet výstupů, pixel je ve stínu

Shadow Volumes



Implementace

- použití *stencil bufferu* (Heidmann, 1991)
- požadavky na geometrii
 - trojúhelníková reprezentace
 - objekty musí být uzavřené (*2-manifold*)
 - musí obsahovat údaje o konektivitě (který polygon sousedí se kterým, např. *Winged-edge*)

Stencil Buffer

- pracuje podobně, jako *z-buffer*
- ukládá celá čísla bez znaménka
- při 24bitovém *z-bufferu* je *stencil buffer* (8 bitů) na NV kartách zadarmo

Stencil Buffer

- *glStencilFunc(func, ref, mask)*
 - *func* - porovnání referenční a uložené hodnoty (*LESS, EQUAL, ...*)
 - *ref* - referenční hodnota
 - *mask* - maskuje před testem jak referenční, tak uloženou hodnotu (*AND*)



Stencil Buffer

- `glStencilOp(fail, zfail, zpass)`
 - *fail* - operace, pokud *stencil test* neprojde
 - *zfail, zpass* - operace v závislosti na výsledku testu v *z-bufferu*
 - operace (*KEEP, ZERO, REPLACE, INCR, DECR, INVERT*)



Postup

1. vykreslení scény s vypnutými světly (pouze ambientní a emisní složka)
2. pro každé světlo
 1. vymazání *stencil bufferu*
 2. vykreslení stínového tělesa (zápis pouze do *stencil bufferu*)
 - a) přední stěny, inkrementace *stencil bufferu*
 - b) zadní stěny, dekrementace *stencil bufferu*
 3. překreslení scény se zapnutým světlem (aditivní blending, *stencil test == 0*)



Postup detailněji

- přední stěny stínového tělesa

```
CullFace(BACK); Enable(CULL_FACE);  
ColorMask(0, 0, 0, 0); DepthMask(0); StencilMask(~0);  
DepthFunc(LEQUAL); Enable(DEPTH_TEST);
```

```
StencilFunc(ALWAYS, 0, ~0);  
StencilOp(KEEP, KEEP, INC); // inkrementace při zpass  
Enable(STENCIL_TEST);
```



Postup detailněji

- zadní stěny (pouze malá změna)

```
CullFace(FRONT);  
StencilOp(KEEP, KEEP, DEC); // dekrementace při zpass
```

- překreslení scény

```
DepthFunc(EQUAL);  
ColorMask(1, 1, 1, 1);  
StencilFunc(EQUAL, 0, ~0); StencilOp(KEEP, KEEP, INC);
```


Zfail Shadow Volumes

- pro uzavřená stínová tělesa je výsledek stejný, jako u *zpass* metody (Carmack [1])
- obě metody je možné kombinovat

Zfail Shadow Volumes

- nastavení *stencil testu* pro stínové těleso
- přední stěny

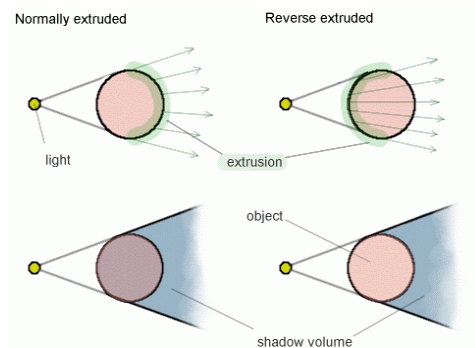
```
StencilOp(KEEP, DECR, KEEP); // dekrementace při zfail
```

- zadní stěny

```
StencilOp(KEEP, INCR, KEEP); // inkrementace při zfail
```

Zfail Shadow Volumes

- uzavření stínového tělesa



Zfail Shadow Volumes

- problém přetrvává
- chybu nezpůsobuje *near*, ale *far* rovina
- tomu lze zabránit...

Posunutí *far* roviny do nekonečna

- o matice klasické perspektivní projekce (*glFrustum*)

$$\mathbf{P} = \begin{bmatrix} \frac{2 \times \text{Near}}{\text{Right} - \text{Left}} & 0 & \frac{\text{Right} + \text{Left}}{\text{Right} - \text{Left}} & 0 \\ 0 & \frac{2 \times \text{Near}}{\text{Top} - \text{Bottom}} & \frac{\text{Top} + \text{Bottom}}{\text{Top} - \text{Bottom}} & 0 \\ 0 & 0 & \frac{\text{Far} + \text{Near}}{\text{Far} - \text{Near}} & -\frac{2 \times \text{Far} \times \text{Near}}{\text{Far} - \text{Near}} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Posunutí *far* roviny do nekonečna

- o nová matice (vypne ořezávání *far* rovinou)

$$\lim_{\text{Far} \rightarrow \infty} \mathbf{P} = \mathbf{P}_{\text{inf}} = \begin{bmatrix} \frac{2 \times \text{Near}}{\text{Right} - \text{Left}} & 0 & \frac{\text{Right} + \text{Left}}{\text{Right} - \text{Left}} & 0 \\ 0 & \frac{2 \times \text{Near}}{\text{Top} - \text{Bottom}} & \frac{\text{Top} + \text{Bottom}}{\text{Top} - \text{Bottom}} & 0 \\ 0 & 0 & -1 & -2 \times \text{Near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Posunutí *far* roviny do nekonečna

- o překvapivě téměř nemění preciznost *z-bufferu* (při *near* 1 a *far* 100 zmenšení pouze o 1%)
- o trik funguje pouze pro perspektivní projekci (ne u ortogonální)

Jak to urychlit

- o *Depth clamping* (*NV_depth_clamp*)
 - GeForce3
 - nedochází k oříznutí fragmentu *near* a *far* rovinou
 - hloubka fragmentu zůstane v intervalu $[\min(z_n, z_f), \max(z_n, z_f)]$
 - není třeba používat speciální \mathbf{P}_{inf} matici a je možné použít ortogonální projekci

Jak to urychlit

- *Two-sided stencil testing* (*EXT_stencil_two_side*)
 - GeForceFX
 - nastavení rozdílných operací pro přední a zadní stěny polygonů
 - stínová tělesa lze vykreslit v jednom kroku

Shadow Mapping



Shadow Mapping, postup

- vyrenderovat scénu z pohledu světla
 - vznikne hloubková mapa (*depth map* nebo *shadow map*)
 - vzdálenost od světla k nejbližším pixelům
- rendering scény z pohledu pozorovatele
 - u každého fragmentu zjistí, jaká je jeho pozice relativně vzhledem ke světlu
 - porovnat vzdálenost fragmentu od světla s hodnotami z hloubkové mapy

Shadow Mapping, test

- vzdálenost fragmentu od světla se **přibližně** rovná údajům v hloubkové mapě
 - fragment je osvětlen
- vzdálenost je větší
 - fragment je ve stínu



Shadow Mapping, vlastnosti

- test se odehrává v *image space*
 - neklade žádné omezení na geometrii ve scéně
 - NURBS, pláty, průhledné textury (*alphatest*)
 - silný alias, chyby
- pouze *spotlights*
- víceprůchodový algoritmus
- RenderMan od Pixaru (ToyStory)



Implementace

- rendering scény z pozice světla
 - naplnění *z-buferu* (žádný blending, osvětlování nebo texturing), pbuffer
 - je třeba použít *PolygonOffset*
 - posun celé scény mírně dozadu, aby polygony nevrhaly stín sami na sebe
 - podrobný popis v [2]
 - *glPolygonOffset(scale = 1.1, bias = 4.0)*
 - výsledek ze *z-bufferu* přesunout do 2D textury
 - *GL_INTENSITY8* formát



Implementace, shadow test

- pozice fragmentu vzhledem ke světlu jde generovat pomocí *eye-linear* texturovacích koordinátů
 - generuje homogenní koordináty (s, t, r, q) , které odpovídají (x, y, z, w) v *light space*, porovnává se r/q s texelem $(s/q, t/q)$
 - *EXT_texture_env_combine* a *alphatest*, (zvládá i TNT)
 - podrobnosti lze nalézt v [2]



HW podpora

- *ARB_depth_texture, ARB_shadow*
 - převzaté SGIX rozšíření
 - GeForce 3 a novější

ARB_shadow

- provádí *shadow test* jako filtr textury
- výsledek testu je 0.0 nebo 1.0
- moduluje barvu výsledkem testu

ARB_depth_texture

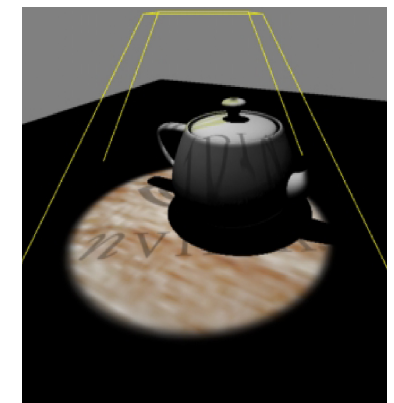
- nový formát textury pro uchování hloubkové informace
 - *GL_DEPTH_COMPONENT16_ARB*,
GL_DEPTH_COMPONENT24_ARB,
GL_DEPTH_COMPONENT32_ARB
- spolupracuje s *glCopySubTexImage2D*
 - optimalizované na NV kartách

Shadow Mapping, poznámky

- nefunguje klasické filtrování *shadow map* textury
 - hloubkové hodnoty nelze zprůměrovat
 - HW musí provést test v každém vzorku a až poté výsledky zprůměrovat
- funguje *mip-mapping*
 - ale nelze použít *gluBuild2DMipmaps*
- podrobnosti v [2]

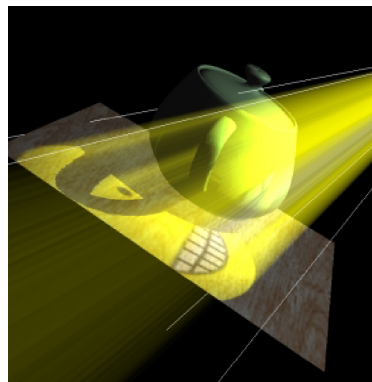
Vylepšení obou technik

- projekce textury
- modulace barvy textury
 - blikající světla



Vylepšení obou technik

- atmosférické efekty



Vylepšení obou technik

- *soft shadows*
 - akumulace více průchodů s rozdílnými pozicemi světla



Odkazy

- [1] John Carmack, *unpublished correspondence*, early 2000.
- [2] Mark J. Kilgard, *Shadow Mapping with Today's OpenGL Hardware*, GDC 2001
- [3] Cass Everitt and Mark J. Kilgard, *Practical and Robust Stenciled Shadow Volumes for Hardware-Accelerated Rendering*, 2002.
- [4] developer.nvidia.com