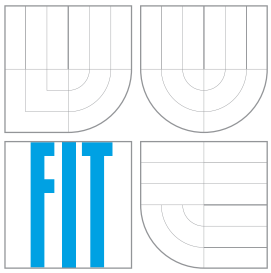
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VIZUALIZACE GRAFICKÝCH SCÉN S VYUŽITÍM STÍNOVÝCH MAP

GRAPHICS SCENE VISUALIZATION USING SHADOW MAPS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ PLACHÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAN PEČIVA, Ph.D.

BRNO 2012

Abstrakt

Cílem této práce bylo seznámit se s technikami pro zobrazování stínů v grafických aplikacích. S hlavním zaměřením na techniky stínových map. Součástí práce byla implementace rozšíření stínových map v demonstrační aplikaci a porovnání jednotlivých technik mezi sebou.

Abstract

The thesis is focused on studying shadow generating techniques in graphics applications with main focus on shadow map techniques. The thesis includes implementation of improvement for shadow map technique in demonstration application and comparing shadowing techniques together.

Klíčová slova

stínové mapy, stíny, aliasing, měkké stíny, ostré stíny, OpenSceneGraph, LiSPSM, PSSM, projekční metoda, vlastní stín, vržený stín, stínová tělesa

Keywords

shadow maps, shadows, aliasing, soft shadows, hard shadows, OpenSceneGraph, LiSPSM, PSSM, projections shadows, self-shadow, cast shadow, shadow volumes

Citace

Tomáš Plachý: Vizualizace grafických scén
s využitím stínových map, bakalářská práce, Brno, FIT VUT v Brně, 2012

Vizualizace grafických scén s využitím stínových map

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jana Pečivy, Ph.D.

.....
Tomáš Plachý
16. května 2012

Poděkování

V rámci této možnosti bych rád poděkoval panu Ing. Janu Pečivovi, Ph.D. za pomoc při vypracování této práce.

Obsah

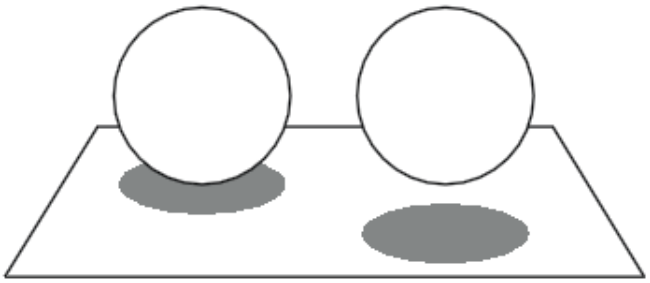
Kapitola 1

Úvod

Stíny jsou nedílnou součástí našeho každodenního života a jejich přítomnost bereme jako samozřejmost. V počítačové grafice však zobrazení stínů a jejich výpočet představuje problém s nelehkým řešením. Tato práce představuje nejpoužívanější techniky pro tvorbu stínů v počítačové grafice, s hlavním zaměřením na techniku stínových map. Kromě vysvětlení principů jednotlivých technik zde naleznete i odpovědi na otázky „proč je důležité stíny zobrazovat" a „jaký má vliv výpočet stínů na výkon aplikace". Součástí práce je demonstrační aplikace, umožnující porovnání vzhledových i výkonnostních rozdílů jednotlivých implementací technik stínových map.

1.1 Význam stínů

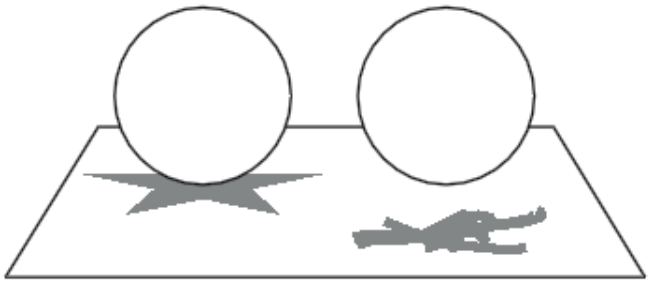
Stíny hrají velkou roli při vytváření realistických trojrozměrných scén a poskytují nám dodatečné informace o objektech. Mohou nám prozradit například jaký je tvar či umístění objektů. Napomohou nám nejen u objektů, které jsou v danou chvíli viditelné, ale i objektů které jsou našemu pohledu skryty. Stín nám například může prozradit, že podlaha není rovná, nýbrž zvlněná. Lze také rozpoznat z jakého směru svítí světlo. [5] [12]



Obrázek 1.1: Vnímání polohy podle stínu [8]

V historii bylo provedeno několik experimentů, které ukázaly význam stínů ve scéně. V roce 1996 Kersten a kolektiv zkoumali vliv stínů na vnímání pohybu. V jednom z experimentů zobrazili kouli nad plochou, podobně jako na obrázku 1.1. Stín v obrázku ovlivňuje vnímání pozice koule. Pokud se stín posune směrem dozadu, pozorovatel má dojem, že koule se pohybuje směrem dozadu a přibližuje se k ploše. [8]

Stejnou informaci o poloze objektů mohou poskytnout i stíny, které neopovídají daným předmětům. Z obrázku 1.2 je vidět, že vržené stíny neodpovídají kouli a přesto je dojem o pozici koule zcela stejný jako na obrázku 1.1. [8]



Obrázek 1.2: Nereálný stín koule [8]

Experimenty provedené v roce 2004 Ni a kolektivem ukazují, že není nutné vytvářet zcela realistické stíny. Pozorovatel si automaticky vytvoří spojení mezi objektem a stínem a akceptuje danný stín. Tento fakt umožnil používání kulatých stínů pod postavami v mnoha starších video hrách.[8]

Z výše uvedených výzkumů tedy vyplívá, že realistické stíny nejsou nutné pro poskytnutí informací o vzhledu a rozložení scény. Nadruhou stranu, pokud budou stíny příliš zjednodušeny, scéna nebude vypadat realisticky. Výpočet věrohodně vypadajících stínů může mít velký dopad na konečný výkon aplikací a správné vyvážení mezi výkonem a vzhledem není jednoduchý úkol.

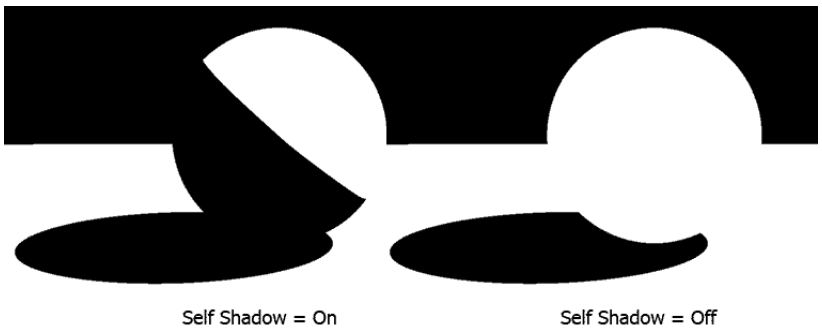
Kapitola 2

Stínové techniky

Narozdíl od reálného světa, ve kterém existuje pouze jeden typ stínů, v počítačové grafice existuje typů hned několik. Tato kapitola popisuje rozdíly mezi jednotlivými typy stínů a vysvětluje principy nejpoužívanějších technik pro jejich výpočet.

2.1 Vlastní a vržený stín

V počítačové grafice se často rozlišují stíny vlastní a vržené. Vržený stín je ten, který vrhá jedno těleso na druhé. Tento stín vnímáme nejčastěji a pomáhá nám rozpoznat vzájemné umístění objektů. Naproti tomu vlastní stín vrhá těleso samo na sebe. Ve vlastním stínu se například nacházejí částí tělesa odvrácené od zdroje světla. Ostínování těchto ploch je řešeno již při fázi osvětlování a nemusíme si ani uvědomovat, že se jedná o stín. Dalším příkladem vlastního stínu je případ, kdy jedna část tělesa vrhá stín na jinou. Některé algoritmy pro generování stínu jsou omezeny pouze na generování vržených stínů. Na obrázku 2.1 je vidět rozdíl mezi koulí s vrženým stínem (vpravo) a koulí na které je vykreslen i vlastní stín (vlevo). [17]

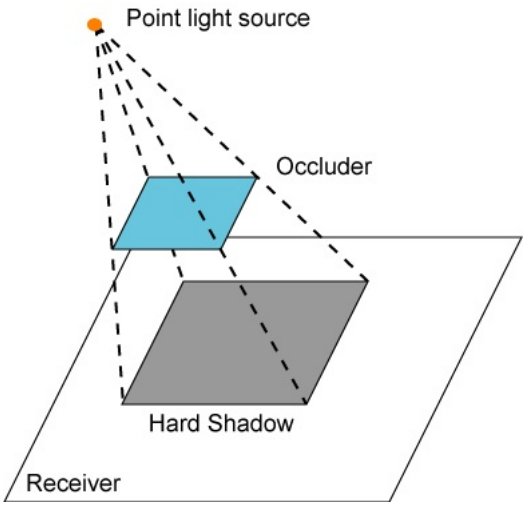


Obrázek 2.1: Ukázka vlastního stínu [16]

2.2 Ostrý a měkký stín

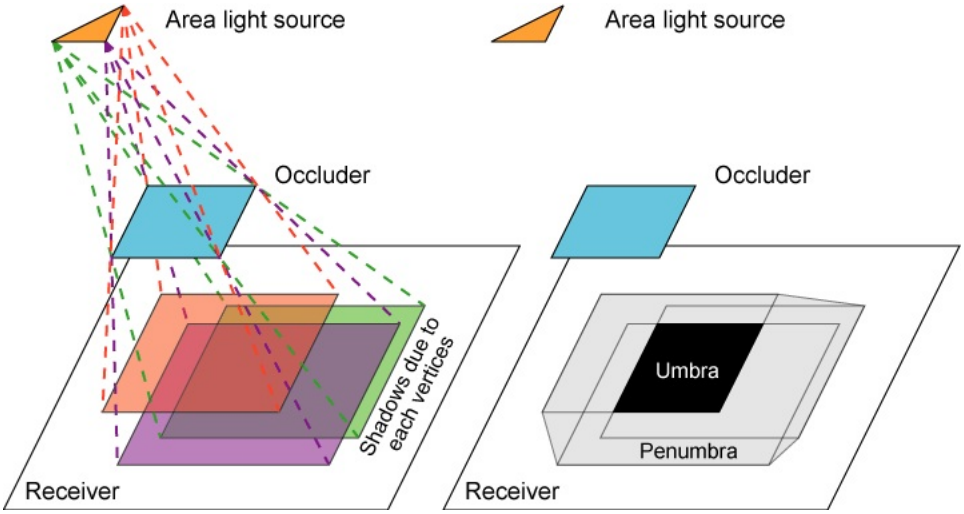
V počítačové grafice se můžeme setkat s pojmy ostré a měkké stíny. Ostré stíny vrhají především bodové zdroje světla. Při vykreslování ostrých stínů existují pouze dvě možnosti.

Na obrázku 2.2 je vidět, že oblast se nachází ve stínu nebo je osvětlená. Ostré stíny jsou jednoduché na výpočet, ale nevypadají zcela realisticky.



Obrázek 2.2: Nákres vzniku ostrého stínu [12]

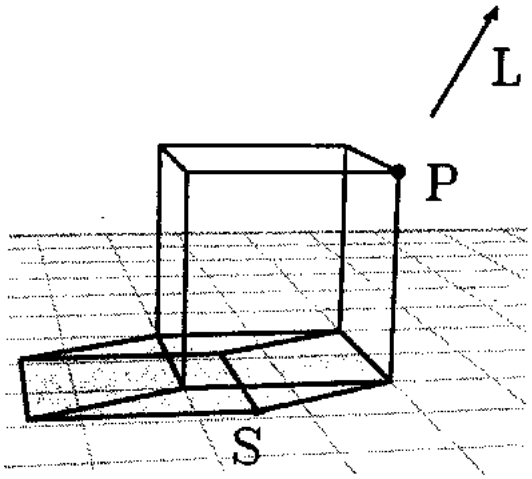
Naproti tomu měkké stíny jsou vrhány především plošnými zdroji světla a měkký stín lze rozdělit na dvě oblasti. První oblast se nazývá Umbra a je to oblast, která je z pohledu světla zcela zastíněna. Umbra je tedy ostrý stín. Druhou oblastí je Penumbra. Penumbra je oblast viditelná z pozice světla jen částečné a nachází se v polostínu. Vznik Umbry a Penumbry je dobře patrný z obrázku 2.3.



Obrázek 2.3: Nákres vzniku měkkého stínu [12]

2.3 Projekční metoda

Projekční metodu představil Jim Blinn v článku „Me and My (Fake) Shadow" [6] v roce 1988. Metoda je určená pro scény s polygonální reprezentací objektů. Pro každou plochu, na kterou bude dopadat stín, je s ohledem na pozici světla nalezana transformace. Tuto transformaci je možné v homogeních souřadnicích popsat projekční maticí 4x4. Pomocí této tranformace je možné zobrazit libovolný objekt ve scéně do roviny dané plochy jako dvourozměrný polygon. Obrázek 2.4 ukazuje příklad zobrazení tělesa do roviny v případě, že světlo L je v nekonečnu. Pomocí projekční matice Mshadow se bod P se zobrazí na bod S. Při zobrazování konečné scény je scéna nejprve vykreslena bez stínů. Poté jsou dokresleny stínové polygony.



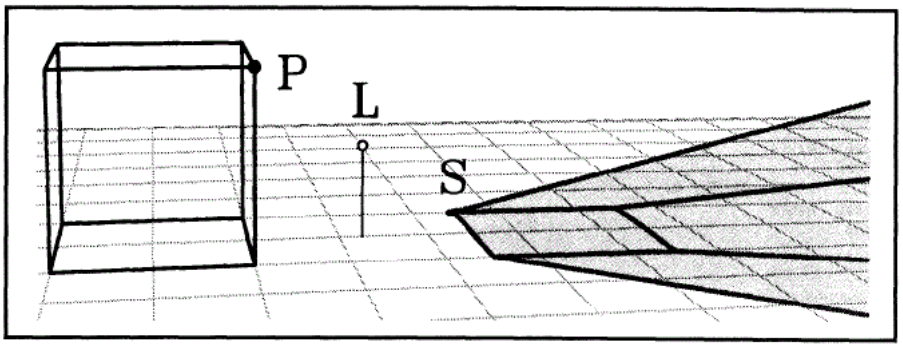
Obrázek 2.4: Projekce tělesa do roviny [6]

Při vykreslování stínů touto metodou se setkáme s několika problémy. Stínový polygon může zasahovat mimo plochu, na kterou je vykreslován. Dalším problémem mohou být matematické nepřesnosti při výpočtech, kdy polygon může prostupovat plochou a ve stínu se následně objeví chyby.

Projekční metoda je velmi jednoduchá, ale trpí několika nedostatky. Protože výsledný stínový polygon je 2D reprezentací původního objektu vrhajícího stín, je nutné aby plocha, na kterou je stín vykreslován, byla částí roviny. Největším nedostatkem této metode zůstává fakt, že nalezaná transformace zobrazuje do dané roviny všechny objekty ve scéně. To vede k zobrazování falešných stínů od objektů, které neleží mezi plochou a světlem a nemohou tedy ovlivňovat osvětlení v daném místě. Obrázek 2.5 ukazuje chybu vykreslení stínu objektu, který se nachází za světlem a neměl by tedy vrhat stín. Existují odvozené techniky pracující s transformacemi odlišným způsobe u nichž falešné stíny nevzdnikají.[17]

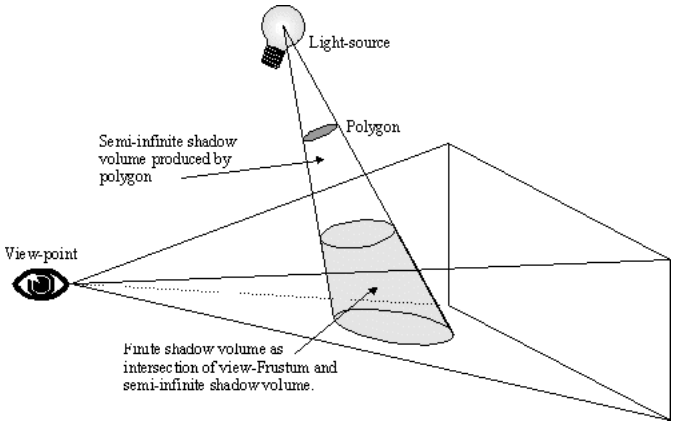
2.4 Stínová tělesa

Stínová tělesa poprvé představil Frank Crow v roce 1977 v článku „Shadow algorithms for computer graphics" [7]. Základní algoritmus pracuje pouze s polygony a bodovými světly, produkuje tedy pouze ostré stíny. Základem je vytvoření tzv. stínového tělesa pro každý z



Obrázek 2.5: Chyba při projekci tělesa do roviny [6]

objektů vrhajících stín. Na obrázku 2.6 je vidět vytvořené stínové těleso, které ohraničuje část prostoru scény. Podobná stínová tělesa je nutno vytvořit pro každý objekt, který vrhá stín.



Obrázek 2.6: Stínová tělesa

Při konečném vykreslování scény se porovnávají pozice jednotlivých objektů scény a stínových těles. Testuje se zda nastane jedna z následujících možností:

- Objekt leží uvnitř stínového tělesa.
- Objekt leží mimo stínové těleso.
- Objekt leží částečně uvnitř stínového tělesa.

V případě, že objekt leží uvnitř tělesa jen částečně, je nutné provést jeho rozdělení na hranici stínového tělesa.

Vytváření stínového tělesa pro samostatný polygon je snadné. V případě obecného objektu je ovšen nutné v prní řadě nalézt jeho obrys, který bude tvořit hranici stínu. Stínové těleso lze vytvořit i bez nalezení jeho obrysu v případě, že vytvoříme samostatná stínová tělesa pro každou jeho plochu přivrácenou ke světlu. Tento postup ovšem znamená větší počet vytvořených stínových těles a tedy i delší dobu výpočtu. [10] [17] [7]

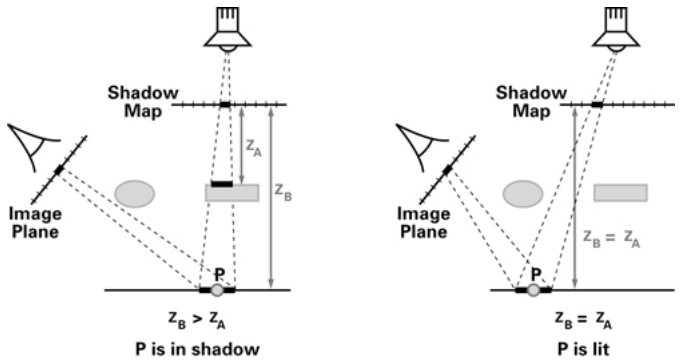
2.5 Stínové mapy

Stínové mapy poprvé představil Lance Williams v roce 1978 v práci nazvané "Casting curved shadows on curved surfaces"[21]. V dnešní době jsou stínové mapy stále objektem výzkumů v počítačové grafice. Je tomu tak zejména pro jejich jednoduchost a nezávislost na složitosti scény. Stínové mapy se dočkali hardwarové implementace, jelikož pro svou realizaci využívají podobný algoritmus, který se používá pro řešení vyditelnosti objektů ve scéně, které je implemetováno hardwarově.

2.5.1 Algoritmus

Technika stínových map funguje na principu, že všechny osvětlené objekty jsou viditelné z pozice světla. Prvním krokem je zjištění objektů viditelných z pozice světla. K tomuto účelu je vytvořen obrázek scény z pozice světla. Tento obrázek je nazýván hloubková či stínová mapa. V každém pixelu mapy není uložena informace o barvě, ale informace o vzdálenosti k nejbližšímu objektu. Ve druhém kroku je provedeno renderování zcény z pozice kamery. Souřadnice pixelů obrazu se transformují do souřadného systému světla a porovná se údaj o hloubce s údajem ve stínové mapě. Je-li hloubka větší, než údaj ve stínové mapě, bod se nachází ve stínu a jeho intenzita se sníží, v opačném případě je bod vykreslen beze změny. Ilustraci obou kroků je zobrazuje 2.7. [18] [8] [17]

1. Zobraz scénu z pohledu světelného zdroje L, hodnoty z paměti hloubky ulož do hloubkové mapy H.
2. Zobraz scénu z pohledu kamery pomocí paměti hloubky.
3. Pro všechny pixely [u, v] (s hloubkou w) zobrazené scény dělej:
   (a) Převeď bod [u, v, w] do soustavy souřadnic zdroje světla L, a získej tak nové souřadnice [x, y ,z].
   (b) A = H[x, y].
   (c) 1. B = z.
   (d) Pokud (A ¡ B), pak je pixel [u, v] ve stínu, jinak je zdrojem L osvětlen.



Obrázek 2.7: Ilustrace kroků algoritmu stínových map

V první kroku algoritmu lze využít tzv. *PolygonOffset*. Před vykreslením scény z pohledu světla se celá scéna posune směrem vzad o hodnotu *PolygonOffsetu*. Posunutím scény předejdeme nežádoucímu efektu, při kterém by polygony mohly vrhat stín sami na sebe.

Fakt, že obak kroky algoritmu využívají standartní rasterizace dává vysoký potenciál pro hardwarovou akceleraci. Například OpenGL poskytuje rožšíření umožnující provedení algoritmu bez zásahu shaderů.

2.5.2 Aliasing

Asiasing je chyba vznikající při diskretizaci spojitých prvků. Aliasing vzniká při nedostatečném počtu vzorků stínové mapy, kdy několik pixelů obrazu připadá na jeden vzorek stínové mapy. V grafické reprezentaci se projeví nejvíce na zkosených hranách, kde hrany nejsou hladké. Obrázek 2.8 ukazuje aliasing na přeponě trojúhelníku způsobený malým rozlišením obrazu. [8] [17]

Obrázek 2.8: Detail aliasingu [14]

Stínové mapy jsou k aliasingu velmi náchylné. Každý zobrazovaný pixel reprezentuje určitou část povrchu těles, ovšem ve stínové mapě může na několik pixelů připadat jedna hodnota. V ideálním případě, by se měl jeden pixel obrazu promítat na jednu hodnotu ve stínové mapě. Stínová mapa je ovšem vytvářena pro celou scénu a u rozhlehlých scén je nutné vytvářet velké stínové mapy. Důsledkem jsou velké nároky na paměť a ztráta výkonu v důsledku velkého množství výpočtů. [8] [17]

Protože aliasing je nejvetším problémem stínových map, bylo vytvořeno množství technik ve snaze ho odtranit. Výsledkem této snahy jsou rozšířené techniky stínových map popsané v následujících kapitolách.

2.5.3 Perspektivní stínové mapy

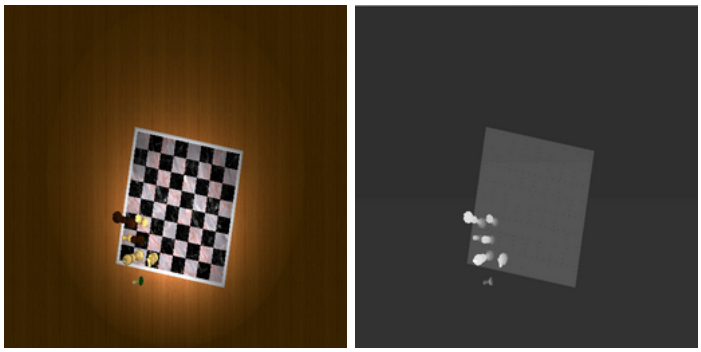
Perspektivní stínové mapy byly představeny na SIGGRAPH v roce 2002 [20]. Tato technika vychází z původních stínových map představených v roce 1978. Narozdíl od původních stínových map je stínová mapa generovnána v post-perspektivním prostoru, díky čemuž je tato technika pohledově závislá. Stínová mapa není generována pro celou scénu, ale pouze pro oblast nacházející se v prostoru pohledu kamery. Díky perspektivní transformaci jsou objekty blíže k pozorovateli větší než objekty vzdálené. Objekty blízké pozorovateli proto mají vyšší počet vzorků v mapě.

Stínová mapa obsahuje pouze informace o právě viditelné části scény. Je tedy nutné generovat novou stínovou mapu pro každý snímek nebo při vetších pohybech kamerou. U

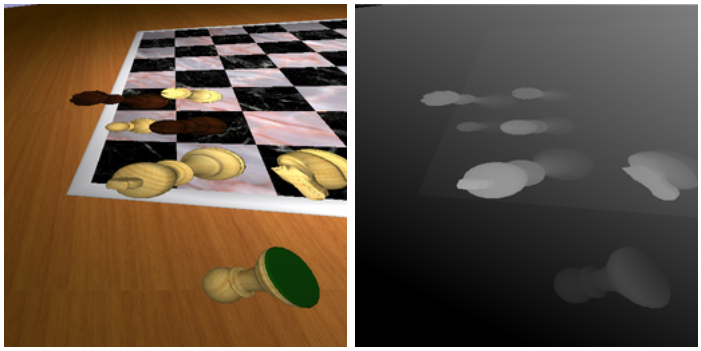
dynamických scén se stínová mapa generuju pro každý snímek znovu z důvodu změn ve scéně. Při použití Perspektivních stínových map dochází pouze k malému poklesu výkonu, způsobeného složitějšími výpočty při tvorbě projekčních matic. [20] [11]

Postup implementace

1. Transformace scény do post-projekčního prostoru.
2. Vytvoření transformace pro světlo
3. Aplikace projekční matice na světlo.
4. Vytvoření stínové mapy.



Obrázek 2.9: Uniformní stínová mapa [19]



Obrázek 2.10: Pespektivní stínová mapa [19]

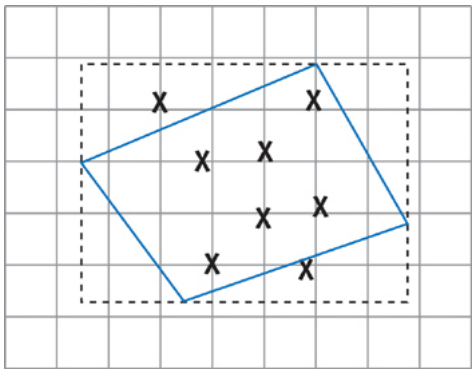
Obrázky 2.9 a 2.10 ukazují pohled světla na scénu a následně získanou stínovou mapu pomocí danné techniky.[11] [19]

2.5.4 Měkké stínové mapy

Technika měkkých stínových map umožnuje generovat měkké stíny i ve scénách s bodovým zdrojem světla. Pomocí techniky PCR (Percentage Closer Filtering) lze nejlépe využít

hardwarovou implementaci stínových map pro generování měkkých stínů v reálném čase.[9] [15] [4]

Základem techniky PCR je provedení několik porovnání pro každý pixel. Konečným výsledkem je pak průměr ziskaných hodnot. Technika počítá kolik procent povrchu se nachází blíže ke světlu a tudíž není ve stínu, podle toho získala svůj název. Algoritmus představil Reeves v roce 1987. Původní algoritmus využíval náhodného vzorkování pro nalezení oblasti ležící ve stínu. První implementace se technika dočkala v renderovacím enginu REYES, ve kterém hledaná oblast byla reprezentována čtyřhranným mikropolygonem jako v obrázku 2.11.[9] [15] [4]



Obrázek 2.11: První implementace Procentuálně blízkého filtrování [9]

Tato technika nám umožňuje generovat měkké stíny s využitím jedné stínové mapy. Při generování ostrých stínů se pro každý pixel získá jeden vzorek ze stínové mapy a výsledke je, zda pixel lezí či neleží ve stínu. Pro generování měkkých stínů se ovšem ze stínové mapy, pomocí souřadnicového posunutí, získá několik vzorků z okolí a jejich kombinací vznikne výsledná hodnota. Tato metoda neprodukuje přesné stíny, ale výsledek je vzhledově shodný s reálnými stíny.[9] [15] [4]

Tradiční metoda pro generování měkkých stínů využívá několika stínových map generovaných z různých pozice plošného zdroje světla a výsledný stín je kombinací vzorků ze všech stínových map. Využití jediné stínové mapy ma za následek zvýšení výkonu. Výpočetní nároky na vytváření několik stínových map jsou nahrazeny méně naročným prohledáváním jediné mapy.[9] [15] [4]

V případě kdy počet vzorků je výrazně menší než velikos penumbry, výsledné stíny mohou vypadat jako několik ostrých stínů poskládaných přes sebe jako v obrázku 2.12. Pro odstranění tohoto jevu je nutné zvýšit počet vzorků získávaných pro každý pixel. Vysoký počet vzorků ovšem může mít za následek výkonnostní problémy.[9] [15] [4]

2.5.5 Paralelně rozdělené stínové mapy

Techniku paralelně rozdělených stínových map (dále jen PSSM) představil Zhang a kolektiv v letech 2006 v práci nazvané „Parallel-Split Shadow Maps for Large-scale Virtual Environments" [22]. PSSM rozdělují prostor pohledu na několik oblastí podle vzdálenosti od kamery. Pro každou z těchto oblastí se generuje samostatná stínová mapa. Tato technika vychází z pozorování, že pro různé vzdálenosti od pozorovatele je nutná jiná hustota vzorků. Díky lepší shodě mezi jednotlivými pixely obrazu a vzorků ve stínové mapě je díky této technice výrazně redukován aliasing.[13]

Obrázek 2.12: Chyba na hraně měkkých stínů [15]

V roce 2001 poprvé Tadamura a kolektiv představil názor na využití několik stínových map. Později v roce 2006 Lloyd a kolektiv využily tento nápad pro implementaci Kaskádových stínových map. Všechny tři výše zmíněné techniky trpí dvěma hlavnímy problémy:

- Jakým způsobem určit hranici pro rozdělení?
- Jak zmírnit propad výkonu způsobený několika renderováními scény při tvorbě stínových map?

Rozdělení pohledu

Vzdálenosti pro rozdělení prostoru pohledu jsou definovány rovnicí 2.1.

$$C_i = \lambda C_i^{log} + (1 - \lambda) C_i^{uni} \tag{2.1}$$

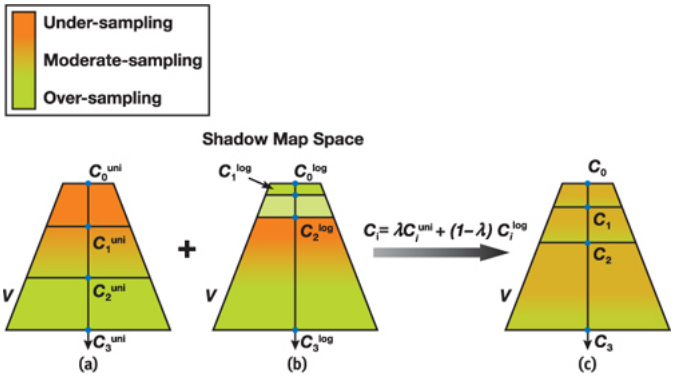
Logaritmické rozdělení C^{log} dává teoreticky optimální rozložení perspektivního aliasingu v celém rozsahu pohledu. Hlavním nedostatkem logaritmického rozdělení je délka jednotlivých části. Části nacházející se blízko u pozorovatele jsou velmi malé a dochází v nich k převzorkování, napopak ve vzdálených částech dochází k podvzorkování 2.13.[13]

Rovnoměrné rozdělení C^{uni} poskytuje stejné rozložení vzorků jako u standartních stínových map. Výsledný efekt je tedy opačný než u logaritmického rozdělení. V částech blízkých pozorovateli dochází k podvzorkování a ve vzdálených k převzorkování 2.13.[13]

V praxi se v PSSM spojuje rovnoměrné a logaritmické rozdělení. Toto spojení produkuje rovnoměrné rozložení vzorků po celé délce pohledu 2.13.[13]

Algoritmus

1. Rozdělení prostoru pohledu na m částí.
2. Vytvoření transformačních matic pro každou z částí.
3. Vytvoření PSSM pro každou z m částí.
4. Sloučení stínů scény.

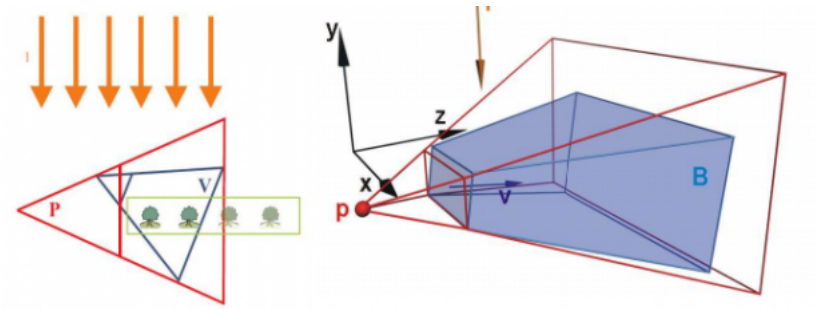


Obrázek 2.13: Rozložení vzorků map

2.5.6 LiSPSM

LiSPSM (Light Space Perspective Shadow Maps) vychází z původních Williamsových stínových map. Cílem LiSPSM je zlepšení kvality stínů využitím perspektivních transformací založených na pozici pozorovatele. Vytváří se perpektivní transformace do prostoru světla.

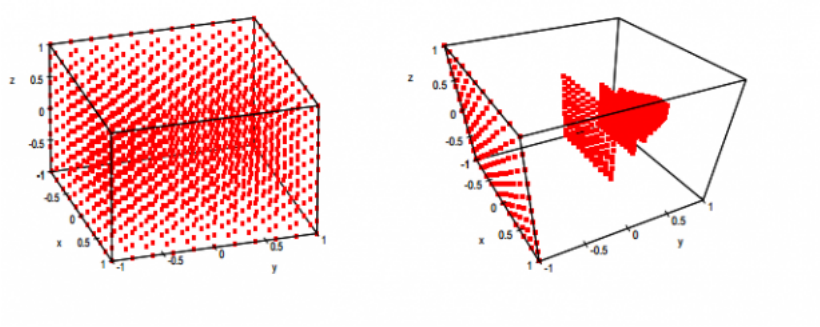
Pro aplikaci transformací se vytváří nová obálka P kolem stávající pohledu kamery B. B obsahuje kromě právě viditelných objektů i objekty, které do pohledu vrhají stín. P má vektor pohledu kolmý na stínovou mapu. V nově vytvořené obálce P lze snadno získat potřebné perspektivní transformace. Směr dopadu světelných paprsků je v obrázku 2.14 značen žlutými šipkami. [3]



Obrázek 2.14: Rozložení vzorků map [3]

Narozdíl od Perspektivních stínových map transformace do prostoru světla nemění směr dopadu světelných paprsků. Směrové zdroje světla zůstávají směrovými a bodové světla jsou zaměněny za směrové. Hlavním cílem perspektivních transformací je změna rozložení vzorků ve stínové mapě.

Světelný prostor je namapován na jednotkovou krychly a kombinací několika matic je vytvořena množina rovnoměrně rozprostřených bodů uvnitř krychle. Jednotlivé body reprezentují světelné paprsky, které mohou dopadat na objekty. Pomocí techniky LiSPSM se pro objekty nacházející se blízko pozorovatele body zhušťují. Výsledkem jsou větší detaily stínů čím blíže se objekt nachází k pozorovateli. Obrázek 2.15 ukazuje rovnoměrné(vlevo) a zhuštěné(vpravo) rozdělení bodů v jednotkové krychly. [3]



Obrázek 2.15: Rozložení vzorků map [3]

Kapitola 3

Demonstrační aplikace

Součástí této práce je demonstrační aplikace implementována pro vizuální a výkonnostní porovnání jednotlivých technik stínových map implementovaných v knihovně OpenSceneGraph (dále jen OSG). Aplikace je napsána v programovacím jazyce C++ s využitím knihovny OSG. Pro vývoj bylo využito prostředí Visual C++ 2008 Express Edition a aplikace je určena pro operační systém Windows.

3.1 OpenSceneGraph

Knihovna OSG je 3D grafická knihovna pro programátory v jazyce C++. Jádro knihovny vuyžívá OpenGL, jinak je knihovna kompletně napsána ve Standard C++. Knihovna nabízí pomocí objektově orientovaného modelu snadnou práci se scénou v podobě stromu, kamerou, světlem a dalšími prvky. Chytré zapouzdření OpenGL setří programátorovi spoustu práce. Na oficiálních stránkách lze naleznout dokumentaci spolu s popsanými ukázkovými příklady a další užitečné informace. [2]

3.1.1 Implementace jednotlivých technik

Demonstrační aplikace využívá různých variací techniky stínových map implementovaných v knihovně OSG. V této sekci jsou popsány některé prvky implementace jednotlivých technik.

Stínové mapy

Implementace Standartních stínových map je v knihovně OSG rozdělena do pěti kroků: [2]

1. Příprava scény pro renderování.
2. Nalezení světla, které bude vrhat stíny do scény.
3. Zaměření kamery z pozice světla do scény pomocí vypočítaných matic.
4. Render scény do textury.
5. Výpočet matic pro přepočet souřadnic scény do stínové mapy.

Měkké stínové mapy

Měkké stínové mapy jsou knohovně OSG implementovány podle metody popsané v GPU Gems 2, kapitola 17 pomocí metody PCF. Postu vytváření stínů pomocí této techniky je podrobně popsán v kapitole 2.5.4. Aby nedocházelo k chybám na hraně stínů jako v obrázku 2.12 je v knihovně implementovám tzv. jittering. [2]

Jittering označuje přidávání náhodného šumu do dat. Přidáním náhodných hodnot na hrany stínů dojde k jejich dalšímu rozmazání a okraje několika stínových map splynou do jednoho celku. Pro použití jitteringu se v knihovně OSG vytváří 3D textura, která je naplněna náhodnými hodnotami. V implementaci popsané v GPU Gems 2 je použit specifický interní formát textury `GL_SIGNED_RGBA_NV`. V knihovně OSG je pro obecnější využití použit formát `GL_RGBA4`, který by měl být nezávyslý na platformě. OpenGLES 1.1 nezná `GL_RGBA4`, proto je v tomto případě použit formát `GL_RGBA`. [2] [15]

Paralelně rozdělené stínové mapy

Algoritmus implementovaný v knihovně OSG vychází z původní myšlenky prezentované v roce 2006 v práci Parallel-split shadow maps for large-scale virtual environments.

Tato metoda rozděluje prostor na několik stínových map, kde bližší předměty mají vyšší hustotu vzorků v mapě. Dalším prvkem, který zmírňuje dopad aliasingu je generování měkkých stínů, které je implementováno pomocí metody PCF. Filtr pro tuto metodu je v knihovně OSG implementován maticí 3.1 o rozměrech 3x3. [2]

1	0	1
0	2	0
1	0	1

Obrázek 3.1: Filtr používaný v PSSM [2]

Impementace techniky PSSM není v knihovně OSG zcela úplná a v aktuální verzi knihovny 3.0.1 plně nepodporuje všechny typy zdrojů světla. Důsledem je vrhání stínů ve scéně kolmo dolů, i když se světlo nachází v jiné pozici nebo kompletní mizení stínů ze scény. Navíc se stíny zobrazují pouze v určitých pozicích kamery. Například při vytvoření směrového zdroje světla došlo k rozmazání stínů po celé scéně. [2]

LiSPSM

Implementace LisPSM vychází z principu posaném v práci z roku 2004 „Light Space Perspective Shadow Maps" [3].

Při výpočtu světelného prostoru vznikají dvě matice `lightview` a `lightproj`. Pokud je ve scéně směrový zdroj světla je obsah obou matic shodny. V případě bodového zdroje světla je v matici `lightview` je uložen otočený snímek z pozice světla a matice `lightproj` obsahuje perspektivní transformaci. Další akce se již odehrávájí ve světelném prostoru, který se otočí tak aby pohled směřoval směrem vzhůru. Dojde k výpočtu matic, které používají směr pohledu jako vektor ukazující vzhůru a pohled se zaměří promítaným směrem. Dalším krokem je namapování prostoru na jednotkovou krychly a rozprostření bodů po kryhli jako na obrázku 2.15.

Původní myšlenka uvedená v práci z roku 2004 naznačuje, že algoritmus by měl fungovat pro všechny typy zdrojů světla. Ovšem implementace v knihovně OSG má problémy s

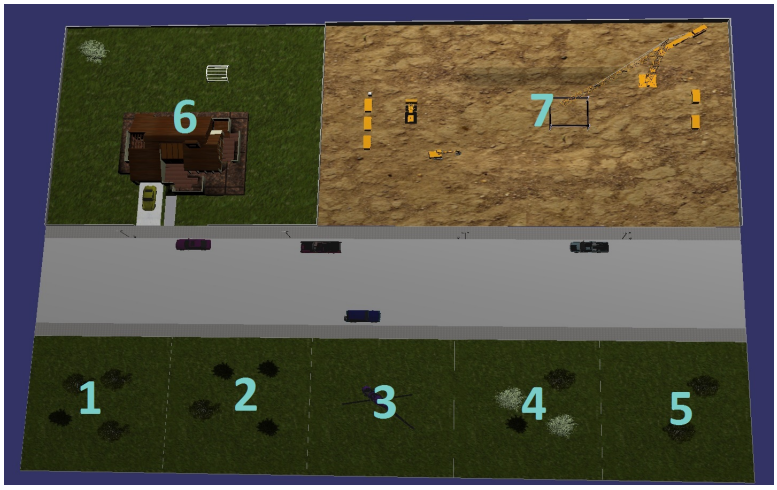
nesměrovými zdroji světla. Důslekem toho je mizení stínů ze scény v případě, kdy se kamera nachází v určité oblasti světelného prostoru. [2]

3.2 Demonstrační scény

Pro demonstraci a srovnání stínových technik byly vytvořeny tři různé scény. Scény se liší svou velikostí i složitostí. Pro vytvoření jednotlivých scén byl použit modelovací program Lexocad a Rhinoceros. Program Lexocad je volně šiřitelný, v případě programu Rhinoceros byla využita studentská license. Objekty použité pro tvorbu scén jsou součástí programu Lexocad, pocházejí ze stránky 3DModelFree.com [1] nebo byly vytvořeny vrámci této práce.

Nejrozlehlejší implementovaná scéne pojmenovaná „Ulice" umožnuje zvolit pro zobrazení stínů kombinovanou techniku. V případě této volby je ve scéně použito několik různých technik najednou. Uživatel tak může porovnat rozdíly mezi těmito technikami bez nutnosti vypínání aplikace. Obrázek 3.2 ukazuje rozložení jednotlivých technik po scéně.

Ve scéně jsou použity celkem tři různé techniky. V oblastech 1-5 jsou použitý běžné stínové mapy. Ovšem v každé oblasti s jiným rozlišením stínové mapy. V oblasti 6 je použita technika měkkých stínových map a poslední oblast 7 využívá techniky LiSPSM.



Obrázek 3.2: Rozložení technik stínových map ve scéně

3.3 Možnosti aplikace

Při spuštění demonstrační aplikace je nutné zvolit scénu, stínovou techniku a způsob ovládání kamery. Aplikace umožňuje dva způsoby volby:

- Jednoduché textové menu.
- Parametry programu.

Pokud je aplikace spuštěna bez parametrů, je uživatel pomocí jednoduchého textového menu vyzván k zadání jednotlivých voleb v následujícím pořadí:

Scéna:

1. Lavička
2. Staveniště
3. Ulice

Stínová technika:

1. Žádné stíny
2. Stínové mapy
3. Měkké stínové mapy
4. Paralelně rozdělené stínové mapy
5. LiSPSM
6. Kombinace technik - tato volba je dostupná pouze pro scénu Ulice

Manipulátor kamery:

1. osg::TrackBallManipulator
2. osg::FirstPersonManipulator
3. osg::OrbitManipulator

V případě volby stínové techniky LiSPSM je uživatel navíc vyzván k zadání počtu kamer na šířku a na výšku obrazu. Pro ostatní techniky nemá tato volba žádný smysl a je vytvořena pouze jedna kamera.

Při zvolení kombinované stínovací techniky je ve scéně ulice použito několik stínových technik. Na různé části scény jsou aplikovány různé stínové technky nebo různá rozlišení stínové mapy. V této scéně pak uživatel vedle sebe vidí odlišnosti jednotlivých technik. Rozložení jednotlivých technik je uvedeno na obrázku 3.2.

Druhou možností volby je zadání pomocí parametrů. Parametry je možné zadat v libovolném pořadí, ale vždy musí být za parametrem následovat mezera a číselná hodnota příslušné volby. Číselné hodnoty voleb jsou stejné jako ve výše zmíněném menu.

Seznam parametrů:

- -sc Volba scény
- -sh Stínová technika
- -m Manipulator kamery
- -cw Počet kamer na šířku
- -ch Počet kamer na výšku

Nastavení aplikace pomocí parametrů bylo implementováno pro možnost vytvoření dávkových souborů. Na přiloženém CD se nachází několik dávkových souborů pro rychlé a jednoduché spouštění různých nastavení demonstrační aplikace.

3.4 Ovládání

Aplikace se spouští v okně s rozlišením 800x600. Pro zapnutí režimu celé obrazovky slouží klávesa *m*. Klávesou *n* lze v aplikaci zapnout zobrazování FPS.

Ovládání kamery se liší v závislosti na zvoleném manipulátoru.

- TrackBallManipulator
  Stisk levého tlačítka myši umožňuje rotovat scénou.
  Stisk pravého tlačítka myši umožňuje scénu přibližovat a oddalovat.

- FirstPersonManipulator
  Při stisku levého tlačítka myši se pohybem myši rotuje kamerou.
  Otočením kolečka se pohybuje kamerou vpřed/vzad.

- OrbitManipulator
  Při stisku levého tlačítka myši se pohybem myši pohybuje kamera po kružnici okolo scény.
  Otočením kolečka nebo stiskem pravého tlačítka a pohybem myši se kamera pohybuje vpřed/vzad.

Pokud jsou stíny ve scéně zobrazeny pomocí techniky stínových map nebo LiSPSM je možné pomocí čísel 1-5 měnit rozlišení stínové mapy za běhu bez nutnosti vypnutí aplikace. Pomocí čísel lze nastavit tato rozlišení stínové mapy:

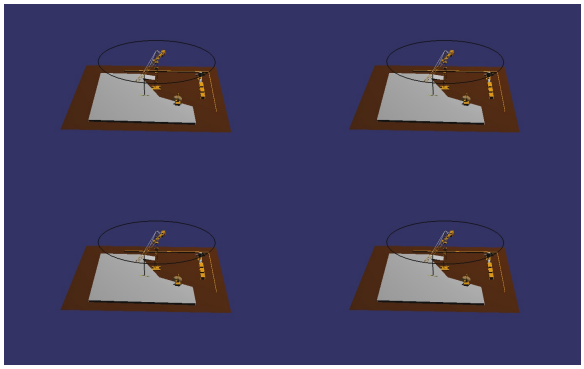
1. 256x256 bodů
2. 512x512 bodů
3. 1024x1024 bodů
4. 2048x2048 bodů
5. 4096x4096 bodů

3.5 Implentované využití LiSPSM

Základní myšlenkou bylo rozdělit obraz na několik menších částí a vykreslit stíny pro jednotlivé části obrazu samostatně. Výsledkem by měla být vyšší kvalita stínů i při použití malého rozlišení stínové mapy. K tomuto účel nelze využít standartní stínové mapy, jelikož stínová mapa se vytváří pro celou scénu a pozice či počet kamer nemají žádný vliv na výslednou kvalitu stínů. Je tedy nutné využít jednu z pohledově závislých technik, v tomto případě byla použita technika LiSPSM.

Pro rozdělení obrazu je nutné vytvořit potřebný počet kamer. V případě rozdělení obrazu na čtyři díly, vytvoříme 4 nové kamery a přidáme je do *osg::viewer* pomocí metody *addSlave()*. Všem kamerám nastavíme stejnou pozici i směr natočení. Každé kameře však přiřadíme pomocí metody *setViewport()* jinou část obrazovky. V tuto chvíli vypadá výsledek jako na obrázku 3.3.

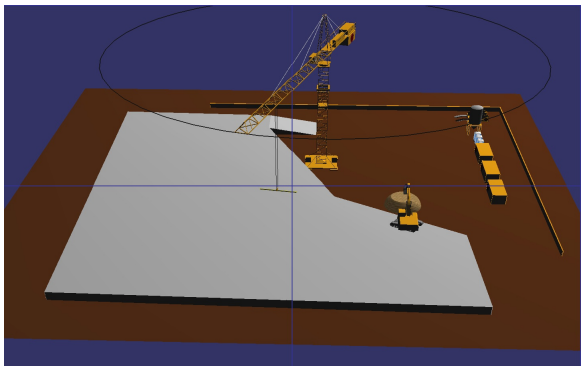
Z obrázku 3.3 je vidět, že každá kamera promítá obraz správně, jen do jí přidělené části obrazovky. Ovšem na všech kamerách je obraz shodný. Obraz se shoduje, protože obraz všech kamer byl vytvořen pomocí stejné projekční matice. Při vytváření každého snímku je vypočítávána nová projekční matice, tato matice se vytváří pro hlavní kameru



Obrázek 3.3: Obraz po vytvoření 4 kamer

a je aplikována na všechny *slave* kamery. Abysme nemuseli v každém kroku ručně počítat projekční matice, pro každou námi vytvořenou kameru, využijeme projekční matici hlavní kamery. Obdobně jako jsme rozdělily obrazovku, nastavíme jednotlivým kamerám pomocí metody *setProjectionMatrixAsFrustum()* jako projekční matici příslušnou část projekční matice hlavní kamery.

V tuto chvíli máme sice správně nastavené všechny kamery, ale obraz stále vypadá jako na obrázku 3.3. Důvodem je implementace metody *UpdateSlaveCallback* pro *slave* kamery. Tato metoda se volá v kažém kroku a aktualizuje hodnoty *slave* kamer v závislosti na hlavní kameře. Tato metoda ovšem v každém kroku nahrazuje projekční matici *slave* kamery za projekční matici hlavní kamery. Posledním krokem je tedy zaregistrování upravené callback metody pro všechny naše kamery, aby námi nastavená projekční matice nebyla vracena do původního stavu. Obrázek 3.4 ukazuje konečný výsledek.



Obrázek 3.4: Obraz složený ze čtyř kamer

Podobného efektu složeného obrazu lze docílit i poskládáním kamer do mřížky místo jejich umístění do stejné pozice. V případě poskládání do mřížky by pak každá kamera smímala celou plochu svého pohledu a ne jen jeho část. Ovšem na přechodech mezi jednotlivými kamerami by docházelo ke zkreslení obrazu, protože každá jeho část by byla snímána z jiného úhlu. Nejvíce by pak tento efekty byl vidět při pohybech scény nebo kamery.

V demonstrační aplikaci je implementována funkce *setupCameras(rowCount, colCount)*. Tato funkce na základě zadaných parametrů, určujících počet kamer na výšku a počet kamer

na šířku, vytvoří při spuštění aplikace požadovaný počet kamer. Následně nastaví všechny výše zmíněné hodnoty, tak aby vznikl plynule navazující složený obraz.

Kapitola 4

Vzájemné porovnání technik stínových map

Tato kapitola obsahuje porovnání výkonosti a paměťové náročnosti jednotlivých technik stínových map implementovaných v knihovně OSG. Současně je zde s těmito technikami porovnáno i implementované využití LiSPSM ve složeném obraze. Z výsledků testování by mělo být jasné, zda je tato implementace použitelná v běžných aplikacích.

4.1 Výkonnostní srovnání

Měření výkonu jednotlivých technik stínových map bylo provedeno pomocí volně dostupné verze programu Fraps. Hodnoty uvedené níže jsou výsledkem meření prováděných v pohybující se scéně po dobu dvaceti sekund. Zkratka FPS (Frames Per Second) reprezentuje počet snímků vykreslených během jedné vteřiny.

Rozlišení mapy v bodech	**FPS**		
	Min	Max	Průměr
bez stínů	124	209	167.05
256x256	70	117	92.05
512x512	75	96	85.35
1024x1024	71	98	86.95
2048x2048	64	100	81.7
4096x4096	50	95	77.5

Tabulka 4.1: FPS v scéně „Lavička" při využití stínových map

V tabulkách 4.1 a 4.2 je znatelný výrazný propad průměrných FPS mezi scénou bez stínů a scénou se stíny. Při zvyšování rozlišení stínové mapy se průměrný počet FPS též snižuje, ale změna není tak patrná jako u scény bez stínů. Tyto malé rozdíly v FPS dokazují rychlost a efektivitu techniky stínových map i pro stínové mapy s velkým rozlišením.

Porovnáme-li mezi sebou jednotlivé techniky stínových map, při stejném rozlišení mapy a na stejné scéně, vznikne tabulka 4.3. Všechna rozšíření původní techniky stínových map se snaží o zmírnění dopadu aliasingu. Složitější algoritmy a další výpočty se ovšem musí projevit na výkonu. Pokud budeme hledat nejvýkonější rozšíření stínových map, pak LiSPSM jsou jasnou volbou.

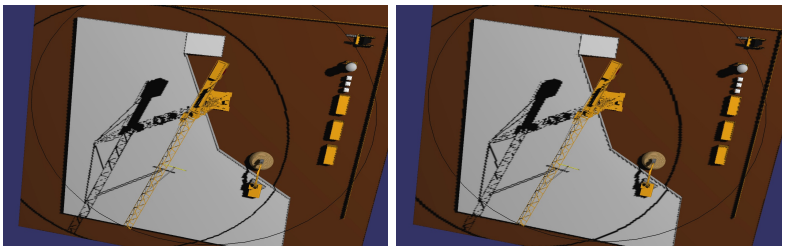
Rozlišení mapy v bodech	FPS		
	Min	Max	Průměr
bez stínů	221	285	243.65
256x256	84	107	97.5
512x512	90	101	95.65
1024x1024	81	95	89.45
2048x2048	75	88	82.55
4096x4096	62	68	65.65

Tabulka 4.2: FPS v scéně „Ulice" při využití stínových map

Rozlišení mapy v bodech	FPS		
	Min	Max	Průměr
bez stínů	221	285	243.65
Stínové mapy	81	95	89.45
Paralelně rozdělené stínové mapy	48	60	55.15
LiSPSM	75	89	83.2
Měkké stínové mapy	42	75	58.35

Tabulka 4.3: Porovnání různých technik na scéně „Ulice" při rozlišení mapy 1024x1024 bodů

Na obrázcích 4.1 je vidět, že pomocí složeného obrazu, můžeme dosáhnout podobné kvality stínů s poloviční rozlišením stínové mapy než při použití jedné kamery. Ovšem i při použití menšího rozlišení stínové mapy má tato implementace velké výkonnostní problémy jak ukazují hodnoty v tabulce 4.4. Ztráta výkonu oproti použití jedné kamery a dvojnásobného rozlišení stínové mapy je 50%, což je příliš velká ztráta.



Obrázek 4.1: Vlevo obraz tvořený jednou kamerou s rozlišením mapy 512x512 bodů, vpravo obraz tvořený čtyřmi kamerami s rozlišením mapy 256x256 bodů

4.2 Paměťová náročnost

Množství využité paměti grafické karty bylo měřeno pomocí utility *Process Explorer* od společnosti Microsoft. Poslední verze této utility je přiložena na CD. *Process Explorer* zobrazuje právě využívanou grafickou pamět s přesností na jedno desetinné místo. Výsledná měření proto nemohou být považována za přesná, ovšem pro potřeby porovnání jednotlivých stínových technik využívaných v této práci je tato přesnost postačující. Naměřené hodnoty

Rozlišení mapy v bodech	Počet kamer		
	1x1	2x2	3x3
256x256	95.4	40.5	17.2
512x512	88.5	39.5	16.3
1024x1024	83.6	37.8	/
2048x2048	78.8	30.5	/
4096x4096	50.3	5.1	/

Tabulka 4.4: Porovnání výkonu při použití více kamer

reprezenutjí rozdíl mezi monožstvím paměti využívaným aplikací bez stínu a množstvím využívaným při zobrazených stínech.

Je zřejmé, že s rostoucím rozlišením stínové mapy poroste i množství potřebné grafické paměti pro její uložení. Ovšem o jak velký nárust se jendná ukazuje tabulka 4.5.

Rozlišení mapy v bodech	Velikost v MB
256x256	3
512x512	7.5
1024x1024	18
2048x2048	66
4096x4096	260

Tabulka 4.5: Velikost stínové mapy v paměti

Hodnoty tabulkách 4.6 a 4.7 ukazují množství grafické paměti využívané pro zobrazení stínů danou technikou při použitém rozlišení stínové mapy 1024x1024 bodů. Hodnoty naměřené pro techniku měkkých stínových map jsou uvedeny pouze orientačně. V knihovně OSG není implementován způsob na změnu rozlišení stínové mapy u této techniky a defaultní hodnota rozlišení není v dokumentaci ani zdrojových kódech uvedena. Dle naměřených výsledků lze usoudit, že rozlišení u měkkých stínových map je nastaveno na 1024 bodů.

Stínová technika	Využitá paměť v MB		
Stínové mapy	15.6	21.1	19.2
Paralelně rozdělené stínové mapy	45.1	51.1	48.3
LiSPSM	18.7	21.2	18
Měkké stínové mapy	16.2	19.4	18.1

Tabulka 4.6: Využití grafické paměti při rozlišení stínové mapy 1024x1024 bodů ve scéně „Lavička”

Z provedených měření je patrné, že použitá technika nemá výrazný vliv na spotřebu grafické paměti. Vyjímku tvoří pouze technika PSSM, kde při rozdělení pohledového prostoru na čtyři oblasti vzrostla spotřeba paměti o 250% oproti ostatním technikám.

Při rozdělení pohledu na více částí dochází s přibívajícím počtem kamer k nárůstu využívané grafické paměti, jak ukazuje tabulka 4.8. Na obrázcích 4.1 je vidět, že pro dosažení

Stínová technika	**Využitá paměť v MB**		
Stínové mapy	16.2	23.1	17.2
Paralelně rozdělené stínové mapy	48	50.3	46.9
LiSPSM	18.2	20.3	17.8
Měkké stínové mapy	12.2	14.4	15.5

Tabulka 4.7: Využití grafické paměti při rozlišení stínové mapy 1024x1024 bodů ve scéně „Staveniště"

Rozlišení mapy v bodech	**Počet kamer**		
	1x1	2x2	3x3
256x256	2.7 MB	4.6 MB	9.7 MB
512x512	5.9 MB	17 MB	37 MB
1024x1024	17.2 MB	65.6 MB	146 MB
2048x2048	66.1 MB	257 MB	352.2 MB
4096x4096	258.9 MB	320.1 MB	400+ MB

Tabulka 4.8: Využitá grafická paměť při zobrazení více kamerami

podobného kvality stínů opradu stačí použít menší rozlišení stínové mapy. Ovšem naměřené hodnoty ukazují témeř shodné množství využité grafické paměti.

Kapitola 5

Závěr

Jak ukazují výsledky prezentované v kapitole 4, neexistuje odpověď na otázku „která technika je ta nejlepší". Každá technika má svoje klady a zápory, které je nutno brát v potaz. Výběr správné techniky, se vždy bude odvíjet od účelu jejího použití. Záleží-li nám na reálnosti zvolíme výkonnostně náročnější, ale vzhledově více se blížící skutečnosti měkké stíny. Ostré stíny nám naopak dovolí rychlé a plynulé zobrazování i v náročných aplikacích.

Původní myšlenkou implemetace složeného obrazu pomocí techniky LiSPSM bylo generování vyšší kvality stínů při menším využití paměti grafické kraty. Tento předpoklad ovšem implementace nesplnila. Stíny vygenerované s nízkým rozlišením stínové mapy mají vyšší kvalitu, než při použití LiSPSM, ale tento úspěch je vykoupen vysokým poklesem výkonu a ke snížení nároků na využití grafické paměti nedošlo. Při upravení algoritmu LiSPSM pro použití tímto způsobem a provedením optimalizace by tento způsob generování stínů mohl v budoucnu být využitelný.

Literatura

[1] *3DModelFree.com* [online]. Dostupné na: <http://www.3dmodelfree.com/3dmodel/list477-1.htm>.

[2] *OpenScenGraph Dokumentace* [online]. [cit. 2012-5-13]. Dostupné na: <http://www.openscenegraph.org/projects/osg>.

[3] *Light Space Perspective Shadow Maps.* [b.m.]: Vienna University of Technology, Austria, 2004.

[4] *Advanced Soft Shadow Mapping Techniques.* [b.m.]: Nvidia, 2008.

[5] Ahokas, T. *Shadow maps.* [b.m.]: Helsinki University of Technology Telecomunications Software and Multimedia Laboratory, 2002.

[6] Blinn, J. Me and My (Fake) Shadow. *IEEE Comput. Graph. Appl.* Leden 1988, roč. 8, č. 1. S. 82–86. Dostupné na: <http://dl.acm.org/citation.cfm?id=44102.44111>. ISSN 0272-1716.

[7] Crow, F. C. Shadow algorithms for computer graphics. *SIGGRAPH Comput. Graph.* červenec 1977, roč. 11, č. 2. S. 242–248. Dostupné na: <http://doi.acm.org/10.1145/965141.563901>. ISSN 0097-8930.

[8] Eisemann, E., Assarsson, U., Schwarz, M. et al. Casting Shadows in Real Time. In *ACM SIGGRAPH ASIA 2009 Courses.* New York, NY, USA: ACM, 2009. SIGGRAPH ASIA '09. Dostupné na: <http://doi.acm.org/10.1145/1665817.1722963>.

[9] Fernando, R. *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics.* [b.m.]: Pearson Higher Education, 2004. ISBN 0321228324.

[10] Holger, W. *Shadow Generation* [online]. [cit. 9. května 2012]. Dostupné na: <http://www.cs.northwestern.edu/ holger/WWWMasters/other/shadow_generation.html>.

[11] King, G. *Shadow Mapping Algorithms.* [b.m.]: NVIDIA Corporation, 2004. Dostupné na: <ftp://download.nvidia.com/developer/presentations/2004/GPU_Jackpot/Shadow_Mapping.pdf>.

[12] LJK, M. L. *Importance of shadow effects* [online]. [cit. 10. května 2012]. Dostupné na: <http://maverick.inria.fr/Research/RealTimeShadows/importance.html>.

[13] Nguyen, H. *Gpu gems 3.* First. [b.m.]: Addison-Wesley Professional, 2007. ISBN 9780321545428.

[14] Nitthilan, K. J. *Anti Aliasing Filter* [online]. [cit. 2012-4-20]. Dostupné na: <http://nittech.blogspot.com/2007/05/anti-aliasing-filter-and-anisotropic.html>.

[15] Pharr, M. a Fernando, R. *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (Gpu Gems)*. [b.m.]: Addison-Wesley Professional, 2005. ISBN 0321335597.

[16] [Puppet]. *Disabling self-shadow* [online]. [cit. !2012-4-2]. Dostupné na: <http://forum.mentalimages.com/showthread.php?4847-disabling-self-shadow>.

[17] Žára, J., Beneš, B. a Felkel, P. *Moderní Počítačová Grafika*. 1st. [b.m.]: Computer Press s.r.o, Brno, 1998. In Czech. ISBN 80-251-0454-0.

[18] Skácel, T. *Realtime stíny v OpenGL*. [b.m.]: Fakulta Informatiky Masarykovy Univerzity, 2003. Dostupné na: <http://www.fi.muni.cz/ sochor/PA158/Slajdy/T4_extra.pdf>.

[19] Stamminger, M. *Perspective Shadow Maps* [online]. [cit. !2012-4-10]. Dostupné na: <http://www-sop.inria.fr/reves/Marc.Stamminger/psm/>.

[20] Stamminger, M. a Drettakis, G. Perspective shadow maps. *ACM Trans. Graph.* červenec 2002, roč. 21, č. 3. S. 557–562. Dostupné na: <http://doi.acm.org/10.1145/566654.566616>. ISSN 0730-0301.

[21] Williams, L. Casting curved shadows on curved surfaces. *SIGGRAPH Comput. Graph.* Srpen 1978, roč. 12, č. 3. S. 270–274. Dostupné na: <http://doi.acm.org/10.1145/965139.807402>. ISSN 0097-8930.

[22] Zhang, F., Sun, H., Xu, L. et al. Parallel-split shadow maps for large-scale virtual environments. In *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*. New York, NY, USA: ACM, 2006. S. 311–318. VRCIA '06. Dostupné na: <http://doi.acm.org/10.1145/1128923.1128975>. ISBN 1-59593-324-7.

Příloha A

Obsah CD

- Bakalářská práce - obsahuje zdrojové kódy LaTeXu
- Demonstrační aplikace - binární soubory aplikace, modely, dávkové soubory pro spuštění
- Zdrojové kódy - zdrojové kódy demonstrační aplikace
- Plakát - plakát ve formátu PDF
- Process Explorer - utilita použitá pro měření vyžité grafické paměťi

Příloha B

Plakát