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

STÍNOVÉ TECHNIKY NA DNEŠNÍM HARDWARE A JEJICH POROVNÁNÍ

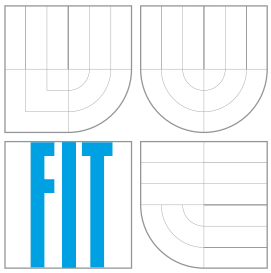
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MICHAL TÓTH

BRNO 2014

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

STÍNOVÉ TECHNIKY NA DNEŠNÍM HARDWARE A JEJICH POROVNÁNÍ

SHADOW TECHNIQUES ON CONTEMPORARY HARDWARE AND THEIR COMPARISON

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MICHAL TÓTH

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAN PEČIVA, Ph.D.

BRNO 2014

Abstrakt

Tato práce se zabývá technikami tvorby stínů ve 3D počítačové grafice. V práci jsou srovnány dvě základní techniky - stínové mapy a stínová tělesa. V práci je také navržena nová technika, kombinující obě předchozí.

Abstract

This term project focuses on basic techniques of creating shadows in 3D computer graphics. Two basic techniques are compared. Those are shadow maps and shadow volumes. Another technique combining previouse two is proposed.

Klíčová slova

C++, Qt, OpenGL, Stíny, Stínové mapy, Stínová tělesa, SSAO

Keywords

C++, Qt, OpenGL, Shadows, Shadow Mapping, Shadow Volumes, SSAO

Citace

Michal Tóth: Stínové techniky na dnešním hardware
a jejich porovnání, diplomová práce, Brno, FIT VUT v Brně, 2014

Stínové techniky na dnešním hardware a jejich porovnání

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením Ing. Jana Pečivy, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Michal Tóth
14. ledna 2014

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Janu Pečivovi, Ph.D. za trpělivost a ochotu při tvorbě této práce.

Obsah

Úvod

Již od zrodu 3D počítačové grafiky je cílem získat obraz nerozeznatelný od reálného světa. K vykreslení co nejreálnějšího obrazu patří, mimo jiné, osvětlení a stíny.

Bohužel, dnešní počítače nemají dostatečný výkon, aby produkovaly tyto obrazy v reálném čase, tj. řádově alespoň desítky snímků za vteřinu. Jeden takový snímek nerozeznatelný od opravdové scény trvá spočítat hodiny až dny.

Pro použití ve vizualizaci dat, strojním návrhu, v počítačových hrách a dalších interaktivních aplikacích je nutné, aby čas potřebný pro tvorbu jednoho snímku scény byl co nejmenší. Pro plynulou animaci je nutné generovat alespoň 25 snímků za vteřinu, aby oko člověka nedokázalo postřehnout přechod mezi snímky a aby animace vypadala plynule. Právě algoritmy pro osvětlení a tvorbu stínů v reálném čase se tato práce zabývá.

Kapitola číslo 1 v krátkosti definuje co je stín, popisuje jeho části a poukazuje na jeho důležitost v obraze.

Kapitola číslo 2 uvádí základní principy tvorby stínů v reálném čase, jejich výhody a nevýhody.

Kapitola číslo 3 se zaměřuje na návrh nové metody kombinující stínové mapy a stínová tělesa k odstranění záporů každé z nich. Na závěr kapitoly je změřena výkonnost všech technik implementovaných v demonstrační aplikaci.

A kapitola číslo 4 popisuje vytvořenou demonstrační aplikaci, detaily implementace a naměřené hodnoty výkonu jednotlivých metod.

Kapitola 1

Stín

Než se začneme zabývat technikami tvorby stínů, je vhodné v krátkosti definovat jak vypadá stín a jak je jeho přítomnost v obraze důležitá.

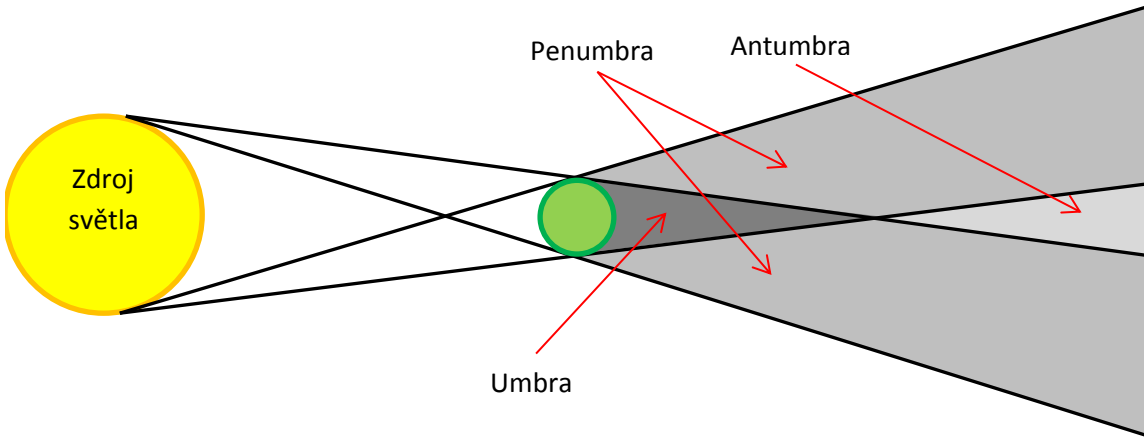
1.1 Druhy stínu

Stín má tři různé oblasti. Každá oblast se liší množstvím dopadajícího světla a tím i jasem. Všechny tři části jsou vidět na obrázku 1.1.

Umbra, plný stín, je nejtmavší část stínu. Zde nedopadá žádné světlo, zdroj světla je kompletně zakrytý.

Penumbra, polostín, je část stínu, kde zdroj světla je zakrytý pouze částečně a je vidět postupný přechod mezi zatemněnou a nezatemněnou částí. Polostín může nastat pouze, když zdroj světla není bodový.

Antumbra je část stínu, ze které je vidět prstencové zatmění zdroje světla. Nastává, pouze pokud je zdroj světla větší než stínící těleso.



Obrázek 1.1: Části stínu.

Následuje několik pojmů používaných v počítačové grafice.

Tvrdý stín je stín obsahující pouze umbru. Tvrdý stín vzniká při použití bodového světla. Většina základních algoritmů pro tvorbu stínů vytváří právě tvrdé stíny.

Měkký stín je stín obsahující všechny tři části. Měkký stín vzniká při použití plošného nebo objemového zdroje světla. Algoritmy pro tvorbu měkkých stínů jsou výpočetně mnohem náročnější.

Vlastní stín je stín zakrývající odvrácenou stranu tělesa.

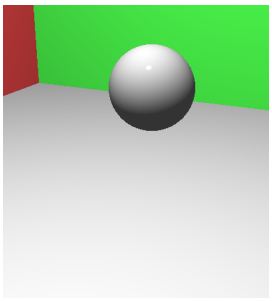
Vržený stín je stín promítnutý na jiné plochy přivrácené ke světlu. Vržený stín vytváří siluetu tělesa.

Pravý stín je stín správného tvaru a velikosti.

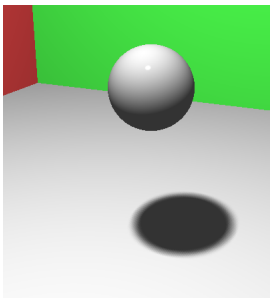
Falešný nebo nepravý stín je stín špatného tvaru. Často jednoduše vytvořená náhrada, např. pouze zatemněný kruh pod tělesem jakéhokoliv tvaru, umístěný kolmo pod těleso nezávisle na pozici zdroje světla.

1.2 Důležitost stínu

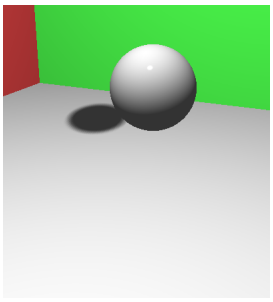
Přítomnost stínu v obraze je velmi důležitá pro chápání rozmístění objektů v obraze. Díky stínu je možné určit vzájemnou velikost a pozici objektů. Ukázka je vidět na obrázku 1.2. Je zde porovnání tělesa (levitující kulička) se stínem a bez stínu. U kuličky bez stínu není možné určit, zda levituje, zda je posazená na podložce, ani vzdálenost od stěn za ní. Pro ilustraci je uveden i příklad, kdy těleso má stín na nesprávném místě a kdy stín má nesprávný tvar. Posunutí stínu mění vnímání pozice tělesa. A změna tvaru stínu zpochybňuje tvar tělesa a mate chápání scény.



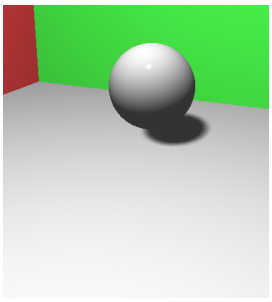
(a) Bez stínu



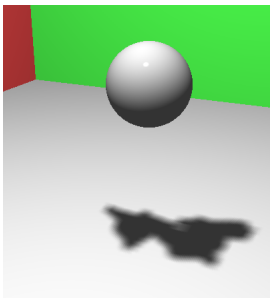
(b) Správný stín



(c) Stín na jiném místě



(d) Stín na jiném místě



(e) Stín špatného tvaru

Obrázek 1.2: Vliv správné pozice a tvaru stínu na vnímání pozice a velikosti objektu. Za předpokladu, že pozice zdroje světla se nemění.

Kapitola 2

Základní stínové techniky

Existuje mnoho způsobů jak tvořit stíny v počítačové grafice. Všechny se liší výpočetní náročností, paměťovou náročností a hlavně kvalitou a správností stínů.

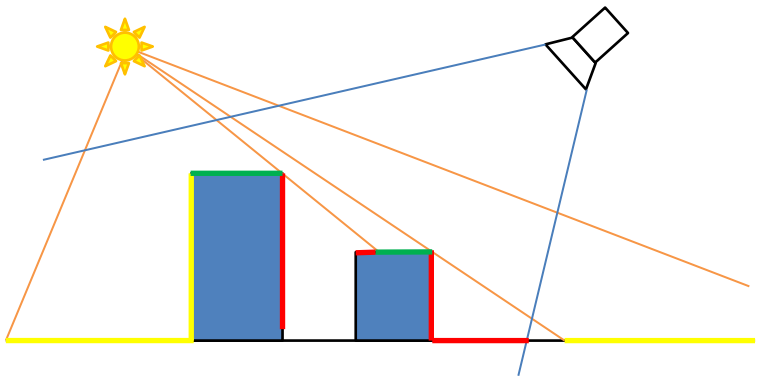
V této kapitole se zaměříme na dvě nejpoužívanější techniky pracující v reálním čase. Jsou to stínové mapy a stínová tělesa. Obě tvoří pravé tvrdé stíny.

Na konci kapitoly se podíváme i na jednu metodu globálního osvětlení – Screen space ambient occlusion.

2.1 Stínové mapy

Techniku stínových map uvedl v roce 1978 Lance Williams v článku *Casting curved shadows on curved surfaces* [5]. Jedná se o jednu z nejrychlejších technik tvorby tvrdých stínů, a proto je také jednou z nejpoužívanějších, převážně v počítačových hrách.

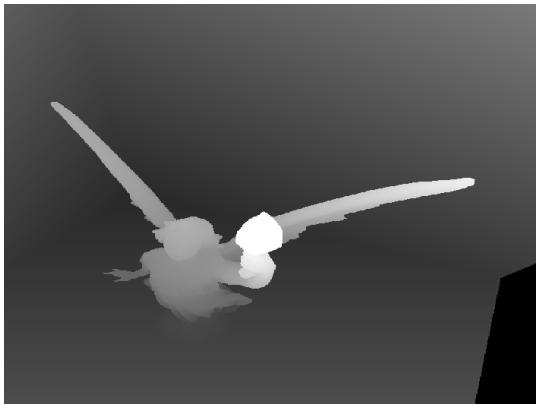
Hlavním principem je porovnávání hloubky vykreslovaného pixelu s hloubkou pixelu z pohledu světla. Pokud se rovnají, pixel je osvícený, jinak je ve stínu. Ilustrace principu viz obrázek 2.1.



Obrázek 2.1: Princip stínových map znázorněný ve 2D. Žluté hrany jsou viditelné pouze z pohledu světla. Červené hrany pouze z pohledu kamery, jsou ve stínu. Zelené hrany jsou viditelní z pohledu kamery i světla a tedy jsou osvícené.

2.1.1 Implementace v grafické kartě

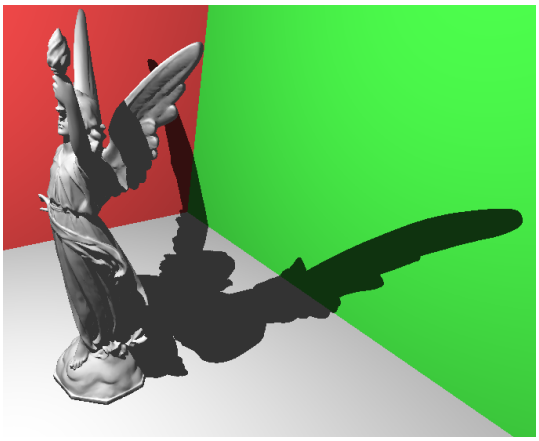
Princip stínových map implementovaný v grafické kartě vyžaduje dva vykreslovací průchody scénou. V prvním průchodu se scéna vykresluje z pohledu světla. V tomto průchodu se ukládá pouze hloubka jednotlivých pixelů do připravené textury (stínová mapa). Druhý průchod vykresluje scénu již z pohledu kamery. Hloubka každého pixelu během tohoto průchodu se testuje s hloubkou uloženou v textuře. Pokud je hloubka vykreslovaného pixelu větší než hloubka získaná z textury, pixel je ve stínu a neaplikuje se na něj osvětlení. Pro správnou funkčnost je nutné transformovat souřadnice pixelu ze světových souřadnic scény do souřadnic v textuře, k tomu se využije transformační matice pohledu světla a matice přepočtu rozsahu displeje na rozsah textury (převod z intervalu <-1,1>na <0,1>). Postup vykreslování je znázorněn na obrázku 2.2



(a) Hloubka z pohledu světla

(b) Hloubka z pohledu kamery

(c) Místa, kde porovnání hloubek selže



(d) Výsledný obrázek

Obrázek 2.2: Postup vykreslování stínů pomocí stínových map. Obrázek (a) je z prvního průchodu scénou. Obrázky (b),(c) a (d) jsou z druhého průchodu scénou.

2.1.2 Problémy

Technika stínových map trpí řadou problémů. Každý z nich lze lépe či hůře řešit. Následuje výčet hlavních problémů.

Směrová světla

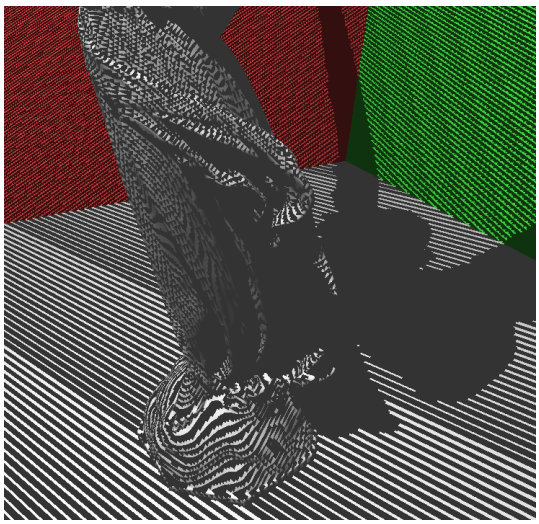
Metoda stínových map z principu funguje pouze u směrových světelných zdrojů. Pokud je potřeba, aby světlo způsobovalo stíny ve všech směrech, je nutné použít šest stínových map pro 6 různých směrů. To přidává další vykreslovací průchody scénou a zvyšuje paměťové nároky.

Sebezastínění

Problém sebezastínění nastává při porovnávání hloubek pixelů, které jsou vyjádřeny desetinným číslem, nejčastěji v intervalu <0,1>. Jelikož čísla s plovoucí řádovou čárkou mají konečnou přesnost, nastává zaokrouhlování při zapisování hloubky do stínové mapy. Tato hodnota se pak v posledním platném místě může lišit od hodnoty hloubky získané z kamery. Tento problém je znázorněn na obrázku 2.3.

Vliv sebezastínění se dá jednoduše odstranit dvěma způsoby. První způsob zahrnuje vykreslování pouze odvrácených stran u vytváření stínové mapy. Tímto se odstraní stín ze sebezastíněných stěn. Je ovšem nutné, aby všechny tělesa byla uzavřená a měla správně nastavené orientace stěn.

Druhý přidává malé posunutí k jedné z hloubek, tím zajistí správný výsledek při porovnávání. Je ale nutné správně nastavit toto posunutí. Pokud je posunutí příliš malé, sebezastínění přetrvá. Ale pokud je příliš velké, odsune stín dál směrem od světla. Toto odsunutí se projeví jako světlý pruh za tělesem, místo zastíněné oblasti.



Obrázek 2.3: Vliv sebezastínění. Nejčastěji se projeví jako pruhy kolmé ke zdroji světla.

Perspektivní a projektivní alias

Problém perspektivního a projektivní aliasu je hlavní nedostatek stínových map. Vychází z různé velikosti a orientace pixelů v pohledech ze světla a z kamery. Problém se projeví jako zuby na hranách.

Lze řešit hrubou silou – zvětšováním rozlišení stínové mapy. Tento způsob problém pouze částečně potlačí, ale za cenu velkého nárůstu potřebné paměti.

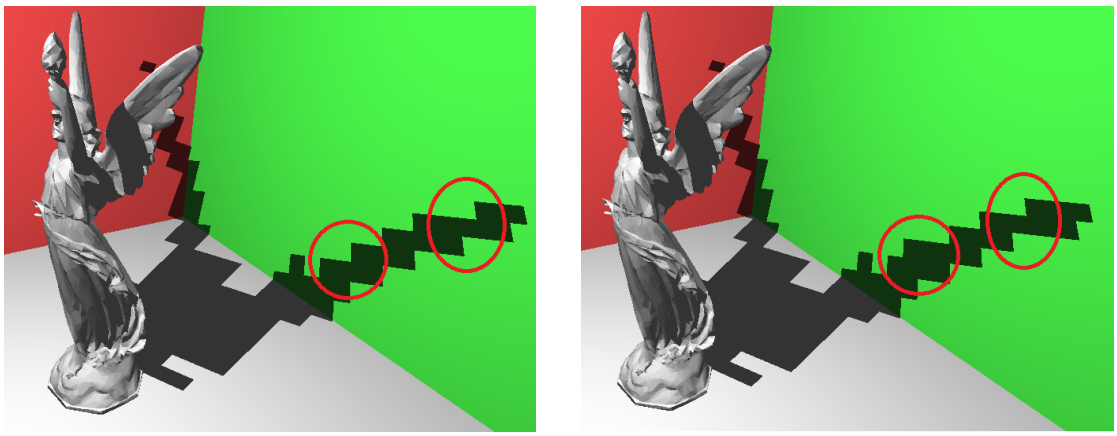
Tento problém je dobře popsán a řešen v práci *Logarithmic prespective shadow maps* [6]. Využívá perpektivní projekci a logaritmickou parametrizaci k odstranění efektu aliasu.

(a) 256x256 (256 kB) (b) 1024x1024 (4 MB) (c) 8192x8192 (256 MB)

Obrázek 2.4: Vliv perspektivního aliasu při různých rozlišení stínové mapy.

Plazení stínu

Problém zvaný *crawling*, česky plazení, je viditelný pouze při pohybu světla a/nebo tělesa. Vychází ze zmíněného perspektivního aliasu. Jedná se o efekt, kdy při pomalém pohybu se objevují a mizí velké kusy stínu.

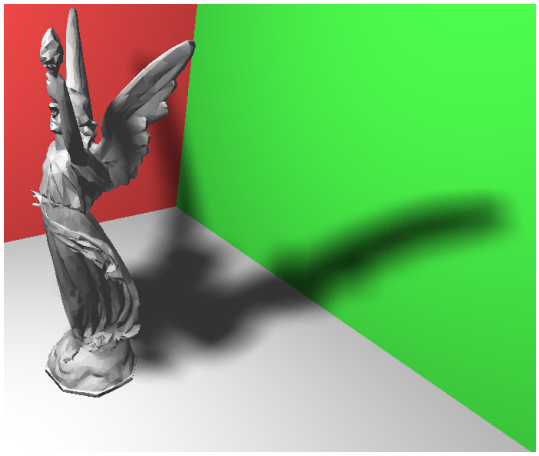


(a) (b)

Obrázek 2.5: Ukázka *crawling* efektu v extrémním případě perspektivního aliasu. Zdroj světla se posunul minimálně, ale změny ve stínu jsou velké.

2.1.3 Vylepšení

Pro stínové mapy existuje mnoho rozšíření řešící různé problémy a vylepšující funkčnost. Zde zmíním pouze jedno z nich, které je i implementované v demonstrační aplikaci k této práci. Jedná se o *Percentage-Closer Filtering* (PCF) navržené ve studiu Pixar [10] [9], toto rozšíření rozmazává hrany stínů a tím vytváří nepravé měkké stíny. Jsou to nepravé stíny, protože stále pracují pouze s bodovým zdrojem světla a neberou v potaz velikost ani tvar zdroje světla. Viz ukázka na obr. 2.6.



Obrázek 2.6: Rozšíření PCF. Tvorba nepravých měkkých stínů.

2.2 Stínová tělesa

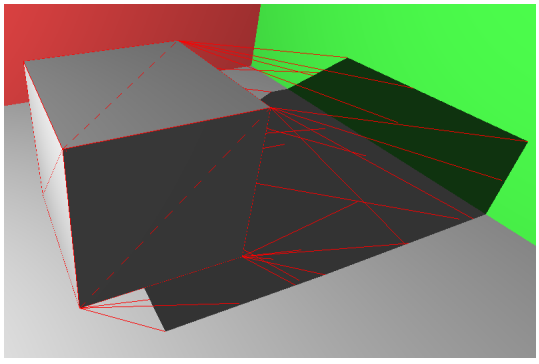
Prostor za objektem vrhajícím stín lze chápat jako objem tělesa a na tomto principu pracuje metoda stínových těles. Pro každý bod obrazu se určí, zda leží uvnitř nebo vně stínového tělesa. Pokud je uvnitř, je zastíněný, jinak je osvětlený. Takto lze vytvořit přesné tvrdé stíny. Tato metoda byla poprvé uvedena v roce 1977 Frakem Crowem[2].

2.2.1 Implementace v grafické kartě

Použití této techniky zahrnuje tři kroky.

Vytvoření stínových těles

Stínová tělesa je nutné vytvořit při každé změně pozice světla nebo při každé změně stínící geometrie (pohyb, změna tvaru,...). Lze je vytvořit částí programu pracující na procesoru, ale pro složitější geometrie je tato operace značně náročná. V demonstrační aplikaci je použit *Geometry shader* z OpenGL pro vytvoření stínového tělesa z každého trojúhelníku geometrie až při kroku vykreslování stínových těles. Použití Geometry shaderu přesouvá zátěž z procesoru do grafické karty. Viz obrázek 2.7.



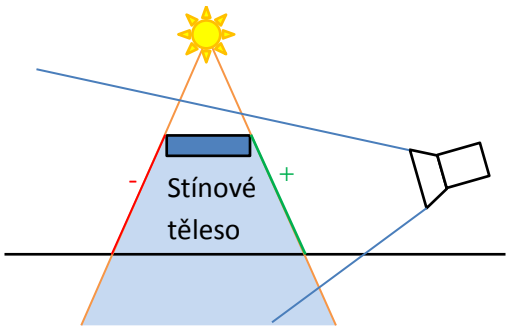
Obrázek 2.7: Ukázka na sebe navazujících stínových těles. Každý trojúhelník vygeneruje jedno těleso.

Vykreslení hloubky scény

V druhém kroku se vykreslí celá scéna, aby se naplnil hloubkový buffer. Zároveň se vykreslí scéna s ambientní složkou světla.

Vykreslení stínových těles

Stínová tělesa se vykreslují s použitím stencil bufferu. Pro každý pixel, který projde hloubkovým testem, se hodnota ve stencil bufferu inkrementuje nebo dekrementuje o 1 v závislosti, zda jde o přivrácenou nebo odvrácenou stěnu od kamery. Viz obrázek 2.8.



Obrázek 2.8: Princip stínových těles. Inkrementace přivrácené a dekrementace odvrácené stěny.

Osvícení nezastíněných míst

Po vykreslení stínových těles v předchozím kroku ve stencil bufferu zbydou hodnoty 0 na místech, které nejsou ve stínu a je potřeba je osvítit. Viz. obrázek 2.9. Osvícení se provede zapnutím stencil testu a vykreslením celé scény pouze s odrazovou a lesklou složkou světla. Výsledný stín na obrázku 2.10.

Obrázek 2.9: Obsah stencil bufferu po vykreslení stínových těles. Černé části jsou nenulové a označují stín.

Obrázek 2.10: Výsledný stín produkovaný stínovými tělesy

2.2.2 Problémy

Ani tato metoda se neobejde bez problémů, uvádíme některé z nich.

Kamera uvnitř stínového tělesa.

Pokud se kamera nachází uvnitř stínového tělesa, všechny stěny tělesa jsou odvrácené a nesprávně tak zapíší do stencil bufferu. Výsledný stín je pak ve většině částí invertovaný, ale ne všude, proto nelze jednoduše stín invertovat zpět.

Tento problém řeší úprava podmínky zápisu do stencil bufferu. Tato úprava se nazývá *depth fail*. Oproti původnímu návrhu (*depth pass*) se do stencil bufferu zapisuje pouze, když hloubkový test selže.

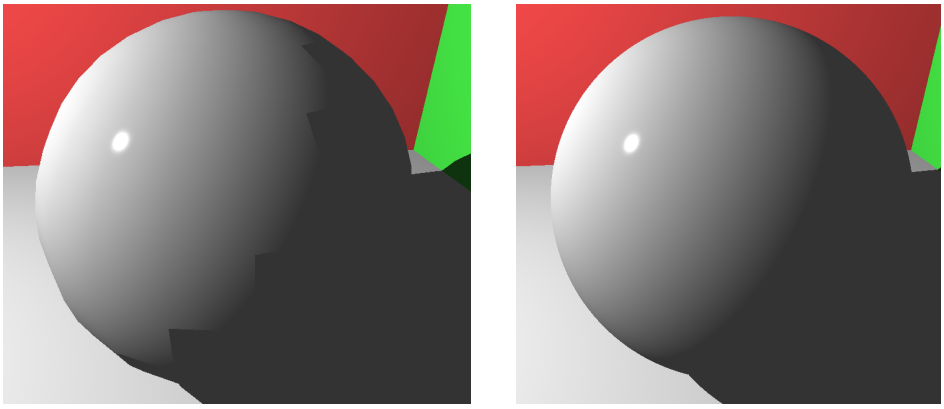
Fillrate

Fillrate, vyplňování trojúhelníků, je největším problémem této metody. Stínová tělesa několikanásobně zvýší počet trojúhelníků ve scéně, tyto nové trojúhelníky navíc mají nekonečnou velikost a vyplňují velkou část obrazovky. Pro představu uvádíme číselné srovnání. V testovacím případě, za použití metody stínových map, bylo vykresleno 400 647 pixelů, ale při použití stínových těles 884 826 899, což je zhruba 22 000-krát více. Toto byl extrémní příklad u modelu s velmi velkým počtem trojúhelníků, jinak se průměr pohybuje okolo 100-násobku.

Tento problém lze řešit oříznutím stínových těles pouze do oblastí, kde mají smysl, toho je využito v *Fast, Practical and Robust Shadows* [8] . Nebo vytvořením stínových těles pouze ze siluet tělesa, místo z každého trojúhelníku. Kapitola 3 se také snaží najít řešení.

Phongovo stínování

Při použití Phongova stínování vznikají artefakty zapříčiněné tím, že Phongovo stínování pracuje po jednotlivých pixelech, ale stíny se tvoří po trojúhelnících. Artefakt se projevuje tmavými zuby na přechodu mezi osvětlenou a zastíněnou částí zaobleného povrchu. Viz obrázek 2.11. Problém lze řešit použitím geometrie s velkým počtem menších trojúhelníků.



(a) (b)

Obrázek 2.11: Artefakty při použití Phonova stínování. Vlevo koule se 760 trojúhelníky, vpravo se 79600 trojúhelníky.

2.3 Screen space ambient occlusion

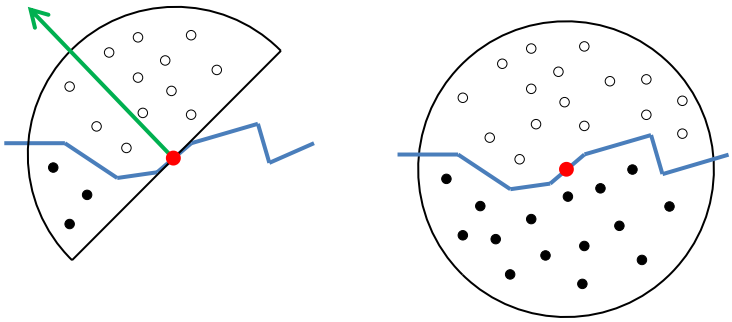
Screen space ambient occlusion (SSAO) není technika přímo používaná ke tvorbě stínů, ale slouží k aproximaci ambientního zastínění. Poprvé byla použita ve hře Crysis [7] z roku 2007.

Obrázek 2.12: Ukázka SSAO z implementace v Crysis od firmy Crytek.

Obraz získaný touto metodou je šedoúrovňová mapa, určující poměr zastínění. Tato mapa se nejčastěji aplikuje na ambientní složku osvětlení, ale lze je použít i jiným způsobem.

Jak již název napovídá, metoda pracuje pouze v prostoru obrazu, místo s kompletní 3D scénou. Pro její funkci je potřeba pouze hloubková mapa uložená v textuře. To umožňuje, aby metoda pracovala s konstantní náročností, nezávisle na složitosti scény. Princip spočívá ve vzorkování pozic náhodných bodů v kouli okolo pozice aktuálního pixelu. Tyto vzorku se promítnou do roviny obrazu a z rozdílu jejich hloubky se počítá vliv na zastínění pixelu.

Takto funguje původní implementace použitá v Crysis. Z důvodu vzorkování v kouli (obr. 2.13) bude pro rovnou plochu polovina vzorků zastíněných a tudíž barva bude šedivá viz. obrázek 2.12.



Obrázek 2.13: Ilustrace vzorkování okolí. Vlevo vzorkování uvnitř polokoule orientovaná normálou, vpravo vzorkování v celé kouli.

Novější metody berou v ohled i normálu aktuálního pixelu a vzorkují z polokoule orientované normálou (obr. 2.13). Toto zajistí, že pro rovné plochy jsou všechny vzorky nezastíněné a tak výsledná barva je bílá. Detaily o vlastní implementaci dále v kapitole 4.3.5.

Kapitola 3

Experimentální technika

Z předchozí kapitoly víme, že stínové mapy jsou velmi rychlá metoda, ale má nepřesné hrany stínů. A také víme, že stínová tělesa jsou přesné, ale za cenu pomalejšího výkonu. Vedoucí práce Jan Pečiva navrhl prověřit metodu, která kombinuje obě.

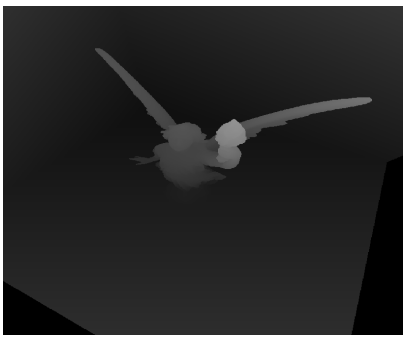
Hlavní myšlenka je v použití stínových map na většinu scény a stínových těles na dodělání hran stínů. Předpokládané urychlení by mělo nastat při vykreslování stínových těles. Stínová tělesa se totiž vykreslují pouze na hranách stínů a nikoliv v celém obraze.

3.1 Postup vykreslení

Tato metoda vyžaduje tři průchody geometrií a jeden průchod nad obdélníkem v prostoru obrazovky. Jednotlivé fáze jsou vidět na obrázcích 3.1, 3.2, 3.3, 3.4.

Stínová mapa

V prvním průchodu se vytvoří hloubková mapa z pohledu světla.

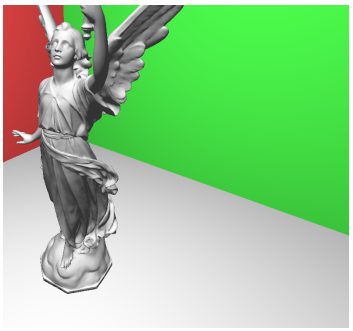


Obrázek 3.1: První průchod – stínová mapa.

Označené hran

V druhém průchodu se podle stínové mapy vytvoří tři obrazy: hrany stínu, obraz bez stínu a obraz se stínem, zvětšeným o jeden pixel stínové mapy. Hrana stínu se označuje do stencil bufferu, zbylé dva obrazy jsou RGB textury.

(a) Hrany stínu ve stencil bufferu



(b) Obraz bez stínu

(c) Obraz se stínem zvětšeným o 1 pixel stínové mapy. Pro ilustraci je zvětšení obarveno červeně.

Obrázek 3.2: Druhý průchod – označení hran.

Vykreslení stínových těles

Ve třetím průchodu se vykreslují stínová tělesa tvořená v geometry shaderu. Vykreslují za použití stencil testu pouze v oblastech okolo hran stínu. Toto zabrání zbytečnému plnění v oblastech, kde není potřeba.

Obrázek 3.3: Třetí průchod – stínová tělesa. Náhled na stencil buffer. Plocha hran je menší oproti obrázku 3.2a.

Doplnění světlých částí

Na závěr se do obrazu se zvětšeným stínem doplní světlé části na hranách, které zbyly ve stencil bufferu.

Obrázek 3.4: Výsledný obraz, po doplnění osvětlených míst na hrany. Identický s metodou stínových těles.

3.2 Zhodnocení výkonnosti

Ani po snaze metodu optimalizovat se nepodařilo dosáhnout očekávané urychlení. Samotná část stínových těles se zrychlyla pouze v rozmezí 4-6%. Přestože stínová tělesa zabírají více než 85% celkového času na snímek, výsledně nepřináší žádné urychlení.

Při změření počtu vykreslených fragmentů (OpenGL dotaz `GL_SAMPLES_PASSED`) byl počet redukován 8 až 30-krát v závislosti na pozici kamery. Toto odpovídá očekávání. Ale přestože fragmenty neprošly stencil testem, stále je bylo nutné rasterizovat, pouze se nespustil fragment shader (který i tak nic nedělá).

3.3 Porovnání s ostatními metodami

Tabulka níže porovnává rychlost technik implementovaných v demonstrační aplikaci. Zajímavý fakt, který lze vyčíst z tabulky je, že rozlišení stínové mapy nemá zásadní vliv na rychlost, pokud je v přijatelné míře (do 2048x2048).

Technika	Rychlost - model 1	Rychlost - model 2
Žádný stín - kontrolní měření	1381,9 fps	345,6 fps
Stínové mapy 256x256	825,7 fps	175,3 fps
Stínové mapy 1024x1024	810,9 fps	173,2 fps
Stínové mapy 2048x2048	756,8 fps	170,0 fps
Stínové mapy 8192x8192	276,3 fps	124,5 fps
Stínové mapy 1024x1024 PCF 5x5	200,9 fps	96,1 fps
Stínové mapy 1024x1024 PCF 9x9	76,1 fps	37,9 fps
Stínová tělesa - depth-fail	53,7 fps	4,6 fps
Stínová tělesa - depth-pass	101,1 fps	6,8 fps
Experiment - 128x128+depth-fail	48,2 fps	4,6 fps
Experiment - 128x128+depth-pass	78,2 fps	6,4 fps
Experiment - 1024x1024+depth-fail	46,4 fps	4,6 fps
Experiment - 1024x1024+depth-pass	72,5 fps	6,3 fps
Experiment - 8192x8192+depth-fail	28,9 fps	4,4 fps
Experiment - 8192x8192+depth-pass	37,2 fps	5,9 fps

Tabulka 3.1: Porovnání rychlosti vykreslovacích metod. Rychlosti jsou uvedeny v počtu snímků za vteřinu. První rychlost je pro testovaný model s 8793 trojúhelníky. Druhá pro model s 157740 trojúhelníky.

Kapitola 4

Demonstrační aplikace

Tato kapitola se zabývá popisem samotné demonstrační aplikace, použité technologie a všech funkcí, které aplikace předvádí.

4.1 Použité technologie

Aplikace je stavěná na multiplatformním frameworku Qt, díky tomu není problém ji spustit jak na systému Windows, tak i na distribucích linuxu. Pro vykreslování 3D grafiky je využito OpenGL ve verzi 4.3.

Vývoj a ladění se provádělo na systému Windows 7 a s grafickou kartou nVidia GeForce GT555M. Pro měření výkonu implementovaných metod byl použit i výkonnější stroj s kartou GTX460 a pro ověření funkčnosti i notebook s kartou od jiného výrobce AMD HD6490M.

4.2 Struktura aplikace

Program je psaný v jazyce C++. Následuje soupis tříd aplikace.

main
: Zaváděcí funkce programu. Vytváří instanci `QApplication` potřebnou pro běh Qt aplikací a zpracovává parametry příkazové řádky. Podporované parametry jsou pouze dva. Parametr "–demo"pro spuštění dema, které automaticky prochází funkce aplikace. A parametr "–record", který uloží každý vygenerovaný snímek do souboru pro pozdější tvorbu videa.

Variables
: Třída pro uchování všech proměnných vztahujících se k uživatelskému rozhraní. Třída odstiňuje uživatelské rozhraní od zbytku aplikace a není tak potřeba posílat signály od prvků GUI přímo k funkcím, které ovlivňují.

GLView
: GLView je třída postavená nad `QGraphicScene`. Slouží pro jednoduché přepínání mezi různými vykreslovacími funkcemi (shadery) a pro odchytávání stisknutých kláves. Tato třída také ověřuje dostupnou verzi OpenGL a případně zahlásí, pokud je dostupná verze nedostatečná.

chess
Pomocné funkce pro `ChessShader` - interaktivní simulace hry šachy.

demo
Třída `DemoController` řídí běh automatického dema – přepíná jednotlivé shadery, ovlivňuje pohled kamery a pozice světel.

scenes
Tato složka obsahuje několik tříd pro stavbu stromu scény a pro načítání 3D modelů.

ModelLoader Třída pro načítání modelů ve formátu .obj do struktury `Model`.

Geometry Třída pro uchování geometrie. Vytváří a naplňuje buffery na grafické kartě.

Material Třída načítání textur ze souborů.

Node Jeden uzel grafu scény. Obsahuje seznam poduzlů, 3D transformace uzlu a případně i geometrii.

Light Struktura pro uložení vlastností světla.

Scene Jedna kompletní grafická scéna. Nabízí funkce pro ulehčení načítání a vykreslování scény.

SceneManager Třída pro správu jednotlivých scény. Vede seznam již načtených scén a umožňuje jednoduše přepínat mezi nimi.

shaders
Složka s jednotlivými vykreslovacími funkcemi (shadery). Každý shader vychází z třídy `BaseShader`, která mu poskytuje základní funkce pro inicializaci a uvolnění OpenGL prostředků, vykreslení scény a vykreslení ladící geometrie – bounding boxy, pozice světel a drátový model. Tato třída se také stará o měření doby vykreslování a výpočet aktuální a průměrné hodnoty FPS (snímky za vteřinu).

4.3 Implementované algoritmy

Zde je výčet všech algoritmů, které jsou v aplikaci implementované. Pro všechny stínové techniky je i uvedené měření výkonu na několika různych modelech o různém počtu polygonů. V grafech a tabulkách jsou uvedené průměrné hodnoty beroucí v potaz různé pohledy kamery a různé úrovně přiblížení.

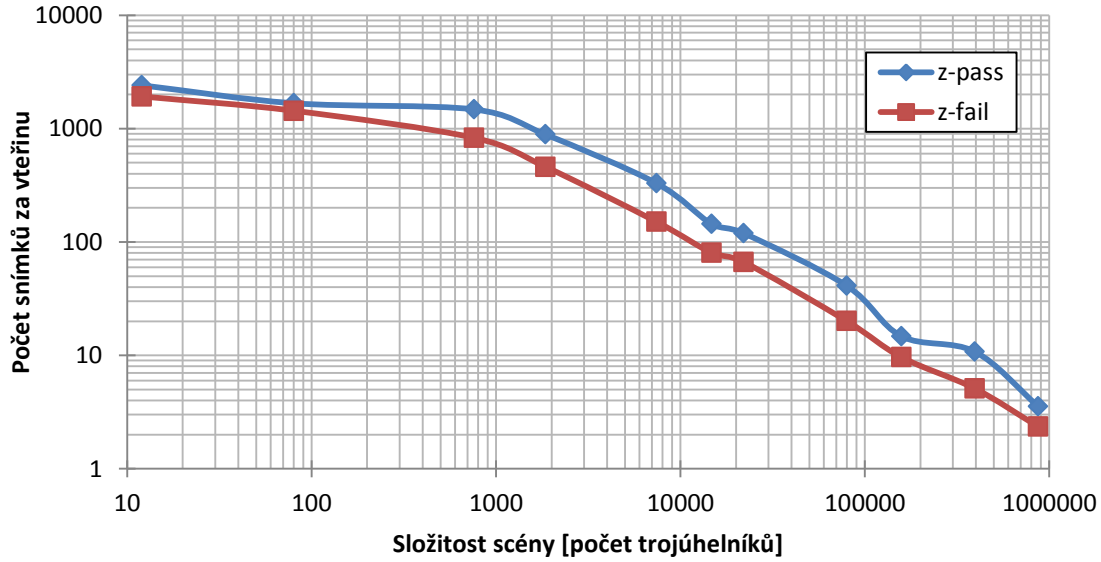
4.3.1 Stínová tělesa

Metoda stínových těles je implementována přesně podle popisu v kapitole 2.2.1. Ke generování stínových těles je použit *geometry shader*, který z každého vstupního trojúhelníku vygeneruje stínové těleso při každém vykreslovaném snímku. Tento přístup je vhodný pro scény s malým počtem trojúhelníků, kde vykreslování stínových těles je poměrně rychlé. Ale pro náročnější scény, kde je velký počet malých polygonů, metoda selhává. Především z důvodu, že se tělesa několikanásobně překrývají a na jeden pixel obrazů se vykresluje i několik stovek stínových těles.

Implementovány byly dvě varianty metody zvané *z-pass* a *z-fail*. Liší se v nastavené hloubkového testu před zápisem do stencil bufferu. Z-pass zapisuje při průchodu testem,

z-fail při selhání testu. Rozdíl mezi metodami je, že *z-pass* selhává v případě, kdy je kamera uvnitř některého stínového tělesa.

V grafu 4.1 je znázorněn výkon obou variant na scénách s různým počtem trojúhelníků. Z měření vyplývá, že pro jednoduché scény (přibližně do jednoho tisíce trojúhelníků) jsou stínová tělesa rychlejší než stínové mapy.



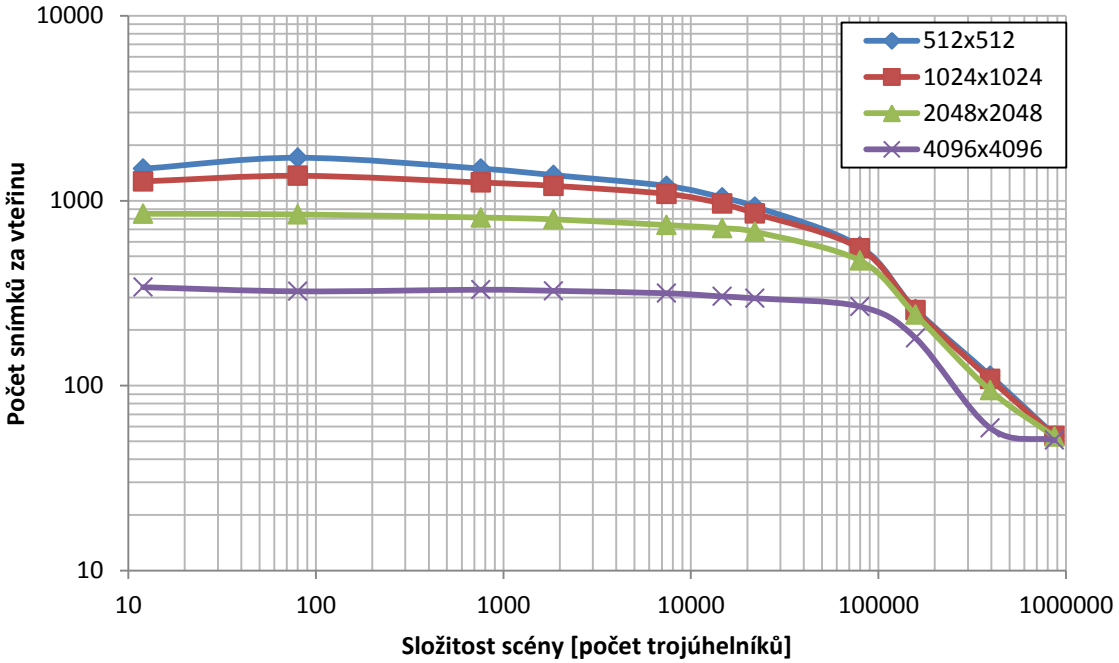
Obrázek 4.1: Graf závislosti výkonu metody stínových těles na složitosti scény

4.3.2 Všesměrové stínové mapy

Implementace stínových map podle principu popsaném v 2.1.1 je funkční pouze pro směrové zdroje světla. Pokud chceme použít bodový zdroj světla, je nutné použít 6 stínových map, každá navzájem otočená o 90°. Proto je nejvhodnější použít texturu ve formátu `cubemap`, což je 6 textur orientovaných na stěny krychle.

Aplikace nabízí několik nastavení pro řešení artefaktů stínových map. Nejdůležitějším parametrem je rozlišení stínové mapy, které i ovlivňuje rychlost vykreslování. Dalším parametrem je offset – odsazení stínu pro potlačení chyby v přesnosti desetinných čísel. Pro uzavřené modely je také možné zapnout vynechání přední strany tělesa a vykreslovat pouze zadní stranu, tím lze většinou odstranit i artefakty sebezastínění.

Graf (4.2) ukazuje rychlost stínových map s různým rozlišením v závislosti na složitosti scény. Z grafu je vidět, že počet trojúhelníků není hlavní faktor omezující výkon metody. Až do prahu okolo sto tisíc trojúhelníků nemělo zvyšování počtu vliv na rychlost. To je dáno především vlastnostmi grafické karty. Zprvu je omezujícím faktorem rychlost zápisu do stínové mapy. Většina testovaných modelů je přibližně stejně velká a zabírá stejně velkou oblast ve stínové mapě, proto se rychlost vytváření stínové mapy ani pozdějšího stínování nemění. Na hranici řádově sto tisíc trojúhelníků se ale začne projevovat druhý omezující faktor a to je rychlost operací nad vertexy a rasterizace, které začnou být náročnější než zápis do stínové mapy.



Obrázek 4.2: Graf závislosti výkonu metody stínových map na složitosti scény

Geometry shader

Jelikož je potřeba vykreslit scénu 6-krát – do každé stěny krychle, stálo za otestování použití geometry shaderu místo volání `glDraw` příkazů 6-krát. Geometry shader umožňuje nastavit do které strany cubemapy se vykresluje. Otestovány byly dvě možnosti. První možnost, v jednom průchodu shaderu vygenerovat 6 trojúhelníků na výstup, každý na jednu stranu. A druhá možnost, nastavit počet volání shaderu (invocations) na 6 a řídit výstup číslem volání (`gl_InvocationID`). Druhá možnost se ukázala jako rychlejší, ale i tak nenabízí žádné zrychlení v porovnání s voláním `glDraw` příkazů pro každou stranu. Při použití geometry shaderu navíc není možné použít ořezávání objektů pro každou stěnu zvlášť a je nutné vykreslovat celou scénu.

PCF

Pro stínové mapy bylo implementováno i rozšíření o metodu PCF, zmíněnou v kapitole 2.1.3, která slouží pro rozmazání hran stínů. Jediným parametrem metody je velikost rozmazávacího filtru. Při použití filtru velikosti 3x3 klesne rychlost vykreslování přibližně na polovinu a dosáhne se jemného rozmazání hran všech stínů, které stačí k zamaskování aliasu na hranách. Pokud chceme dostat měkké stíny, které vypadají reálněji, musíme použít filtr alespoň 9x9. Jelikož se jedná o metodu využívající hrubou sílu ke sčítání oblasti o velikosti filtru pro každý pixel, je pro větší filtry značně pomalá. Pro filtr 9x9 byla rychlost vykreslování méně než desetina v porovnání s metodou bez filtru.

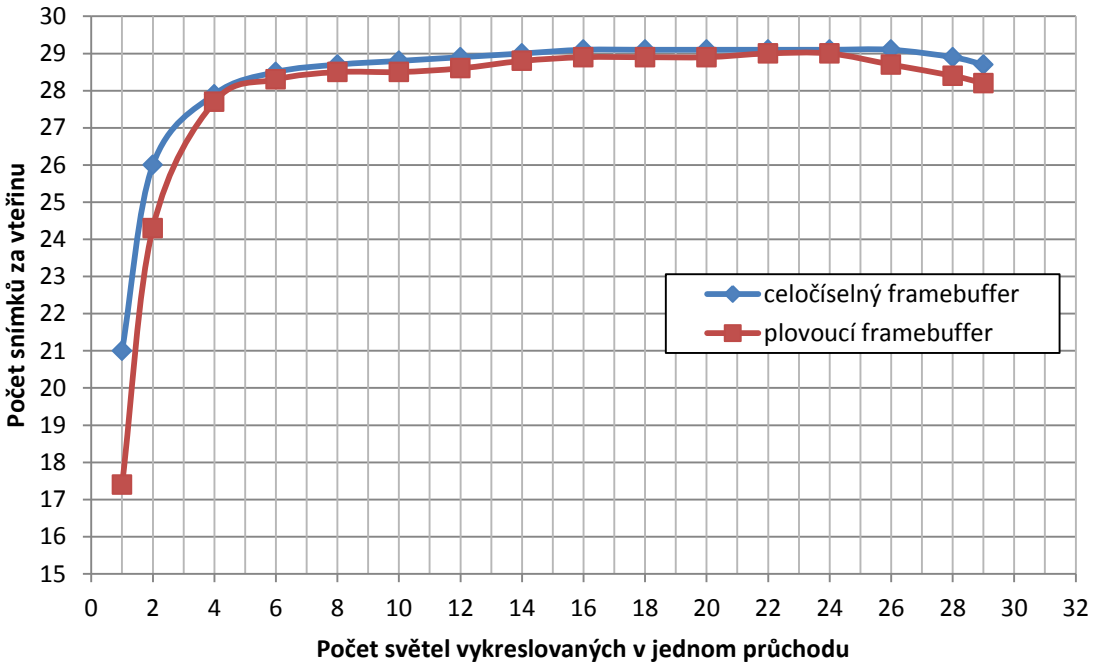
4.3.3 Stínové mapy pro více světelných zdrojů

Zřídkakdy stačí pouze jeden světelný zdroj pro složitější scény, proto se tato práce i povrchově zabývá problémy při vykreslování stínů od více světel.

Hlavním problémem je nutnost vykreslit scénu do cubemapy pro každé světlo a následně přidat vliv osvětlení do výsledného obrazu. Zaměříme se tu až na poslední část, kdy se akumuluje vliv všech světel na osvětlení daného pixelu.

Snaha byla počítat vliv na osvětlené od co nejvíce světel najednou, aby se ušetřily zápisy do paměti. Ideální by bylo projít všechna světla v jednom průchodu, ale je tu jisté omezení. Každé světlo potřebuje, kromě barvy a dosahu, také texturu – stínovou mapu připojenou do texturovací jednotky. Grafická karta, na které bylo prováděno měření, jich má pouze 32, z nichž jsou 3 rezervovány na textury normály, pozice a materiálu. To umožňuje vykreslovat maximálně 29 světel v jednom průchodu.

Za zmínku také stojí, že pokud osvětlení ve scéně se dosahuje velkým počtem málo intenzivních světel, je běžný framebuffer s 8 bity na kanál nedostačující. Jedno samotné světlo pak má vliv na osvětlení tak malý, že se vlivem přesnosti čísel zaokrouhlí na nulu. Vliv zaokrouhlování se dá potlačit zmíněným vykreslováním více světel najednou, kde se vnitřně počítá s čísly s plovoucí desetinnou čárkou a až suma se přičítá do celočíselného framebufferu. Nebo můžeme použít framebuffer s plovoucí desetinnou čárkou, který ovšem zabírá čtyřikrát více paměti a má pomalejší zápis. Graf 4.3 ukazuje výkon v závislosti na počtu vykreslovaných světel na snímek.



Obrázek 4.3: Graf výkonu vykreslování v závislosti na počtu světel na jeden průchod

Simulace plošného zdroje světla

Pomocí velkého počtu bodových světel se dá simulovat i plošný zdroj světla. Nejedná se o nejrychlejší způsob, ale umožňuje vytvářet měkké stíny, které jsou na rozdíl od PCF správné. Na obrázku 4.4 je vidět tento měkký stín v porovnání se standardní stínovou mapou a s PCF.

(a) Standardní stínová mapa

(b) Stínová mapa s PCF

(c) Simulace plošného zdroje světla.

Obrázek 4.4: Porovnání tvrdého stínu standardní stínové mapy a měkkých stínů vzniklých metodou PCF a simulací plošného zdroje světla

4.3.4 Hybridní metoda

V této práci navržená hybridní metoda má vyhrazenou celou kapitolu 3. Zde je uvedená jen pro úplnost.

4.3.5 SSAO

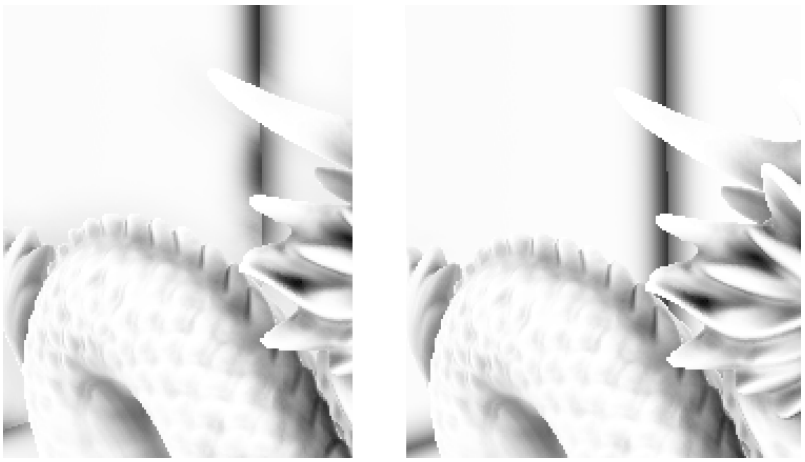
Základ vlastní implementace SSAO vychází z článku Johna Chapmana [4]. Pro každý pixel se vzorkuje náhodné kruhové okolí. Z hloubky každého vzorky se spočítá původní pozice ve světových souřadnicích. Faktor zastínění se oproti původní metodě od Cryteku, kde se porovnávaly jen hloubky, počítá jako skalární součin normály a směru z pozice vzorku do pozice aktuálního pixelu. Díky tomu není nutné transformovat vzorkovací polokouli podle normály, čehož využívají některé metody.

Metoda je upravená tak, aby vyřešila problém, který vychází z faktu, že metoda funguje pouze v prostoru obrazovky a nemá informaci o celé 3D scéně. Problém se projevuje jako černá zář okolo objektů (obr. 4.5). Zář vzniká na místech, kde je velký rozdíl hloubek vedle sebe (např. například obrys postavy, která stojí před zdí) a je zapříčiněná tím, že při výpočtu faktoru zastínění pro pixely na zdi se vzorkují i pixely na postavě, které jsou ale mnohem blíž ke kameře a počítají se jako zastiňující.

Obrázek 4.5: Efekt záře u SSAO

Prvním řešením problému je, že se vliv vzorků na zastínění snižuje se vzdáleností od aktuálního pixelu. Tímto se zář výrazně potlačí, ale stále se tu vyskytuje a vizuálně rušivá. Dalším řešením je navíc přidat kontrolu vzdálenosti. Při překročení nastaveného prahu se vzorek zahodí a nepočítá se. Zář se tak kompletně odstraní, ale pouze za předpokladu, že vznikala na rovné ploše. Pokud vznikala například na rohu místnosti, projeví se jako zář opačné barvy. Na obrázku 4.6 vlevo je vidět, jak okolo rohů vzniká bílá zář.

Finální řešení vychází z metody *depth peeling* [1]. Místo pouze jedné hloubkové mapy se vytvoří několik vrstev. Takže v případě, že vzorek neprojde testem na vzdálenost, je přečtena hloubka z další vrstvy. Pro jednodušší scény stačí pouze dvě vrstvy k odstranění záře.



Obrázek 4.6: Vlevo efekt záře potlačený testem vzdálenosti. Vpravo kompletně odstraněná zář.

Obrázek 4.7: Ukázka vypočteného zastiňovacího faktoru s řešeným problémem záře

4.3.6 Materiály

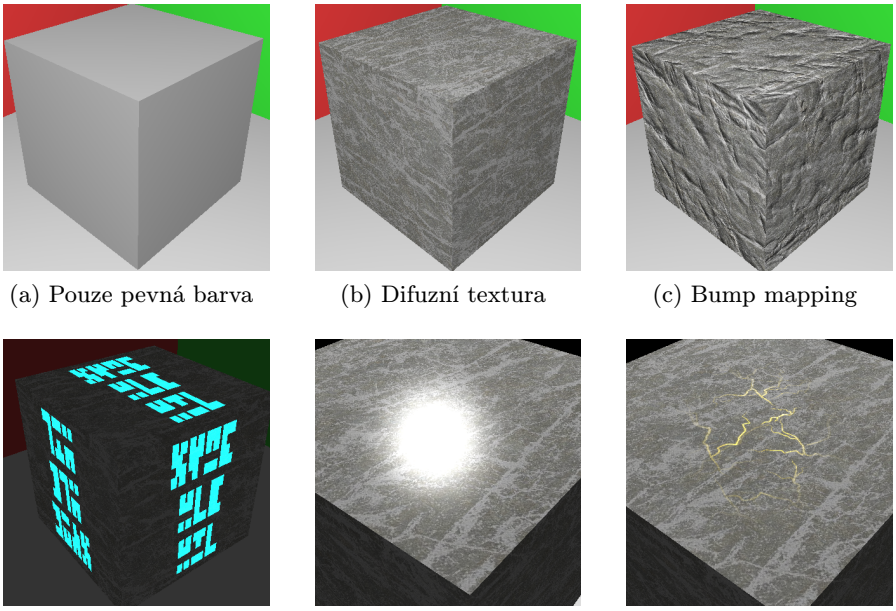
Součástí zadání práce bylo i implementovat materiálový model. Jako materiál chápeme sadu vlastností povrchu. Tyto vlastnosti nemusí být pro celý povrch konstantní, a proto většinou mají formu textur. Uvažujeme následující vlastnosti.

Ambientní a difuzní barva je základní barva povrchu tělesa, tato barva interaguje s ambientní a difuzní složkou světla. Lze je uvést a modifikovat i nezávisle na sobě, ale často to bývá stejná barva.

Spekulární barva ovlivňuje barvu odlesků na povrchu. Toho lze využít ke tvorbě efektů, kde se leskne jen část tělesa, nebo ke změně stylu odlesku. Viz příklad níže.

Emitovaná barva je vlastní barva vyzařovaná povrchem, nezávisle na přítomnosti ostatních zdrojů světla. Nejedná se o zdroj světla, takže nemůže osvítit ostatní předměty. Tato vlastnost je nejvhodnější pro efekty typu podsvícení, různé neony, svící kontrolky a ostatní prvky pro které stačí, aby osvítili pouze sami sebe.

Normálová mapa je textura, která modifikuje normálu získanou z 3D modelu. Lze tak dosáhnout iluze hrbolatého povrchu. Tento způsob změny povrchu vychází z [3] a říká se mu *bump mapping*.



(a) Pouze pevná barva (b) Difuzní textura (c) Bump mapping

(d) Emitovaná barva (e) Běžný odlesk ze světla (f) Odlesk modifikovaný spekulární texturou

Obrázek 4.8: Ukázka několika materiálových vlastností a efektů

Kapitola 5

Závěr

Tato práce pojednává o základních technikách pro tvoru stínů v 3D počítačové grafice a uvádí jejich problémy a možná řešení. Soustředí především na metodu stínových map a metodu stínových těles, ale zabíhá i trochu do oblasti globálního osvětlení v podobě Scree space ambient occlusion.

Během této práce byla navržena metoda, která kombinuje stínová tělesa a stínové mapy s příslibem urychlení vykreslování v porovnání se samotnou metodou stínových těles. Bohužel se ukázalo, že metoda nenaplňuje očekávání a přináší pouze malé, skoro žádné urychlení. Avšak byl ukázán způsob rozdělení výpočtu stínů na oblasti rychle spočitatelné a na oblasti, které je nutné spočítat přesně silnější metodou.

Vzniklá demonstrační aplikace implementuje všechny zmíněné stínové techniky. S pomocí aplikace byly metody proměřeny a byla porovnávána jejich rychlost. Aplikaci lze snadno rozšiřovat a bude sloužit jako základ pro budoucí experimenty.

Literatura

[1] Cass Everitt. *Interactive Order-Independent Transparency.* Nvidia, 2001.

[2] Franklin C. Crow. Shadow Algorithms for Computer Graphics. *ACM SIGGRAPH Computer Graphics.* 1977, roč. 11. S. 242–248.

[3] James F. Blinn. Simulation of wrinkled surfaces. *ACM SIGGRAPH Computer Graphics.* 1978, roč. 12. S. 286–292.

[4] John Chapman. *SSAO Tutorial* [online]. 2011-01-05 [cit. 2014-05-14]. Dostupné na: <`http://john-chapman-graphics.blogspot.com/2013/01/ssao-tutorial.html`>.

[5] Lance Williams. Casting curved shadows on curved surfaces. *ACM SIGGRAPH Computer Graphics.* 1978, roč. 12. S. 270–274.

[6] Lloyd, D. B. *Logarithmic perspective shadow maps.* Chapel Hill, NC, 2007.

[7] Martin Mittring. Finding Next Gen – CryEngine 2. *ACM SIGGRAPH Computer Graphics.* 2007. S. 97–121.

[8] McGuire, M., Hughes, J. F., Egan, K. et al. *Fast, Practical and Robust Shadows.* Austin, TX: NVIDIA Corporation, 2003.

[9] Randima Fernando. *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics.* 1st edition. Boston, US: Addison-Wesley, 2004. ISBN 0-321-22832-4.

[10] William T. Reeves, David H. Salesint, Robert L. Cook. Rendering Antialiased Shadows with Depth Maps. *ACM SIGGRAPH Computer Graphics.* 1987, roč. 21. S. 283–291.