Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE

Štěpán Hrbek

Radiosita v dynamických scénách

Katedra software a výuky informatiky
Vedoucí diplomové práce: doc. ing. Jiří Žára, Csc.
Studijní obor: Informatika

Děkuji Danielu Sýkorovi, který mě celý rok neúnavně podporoval a svým entuziasmem poháněl vpřed, a bez kterého by vznikla jen další diplomová práce pro dva čtenáře.

Diplomovou práci jsem vypracoval samostatně s použitím citovaných pramenů. Souhlasím s jejím zapůjčováním.

V Praze dne 10. prosince 2000 Štěpán Hrbek

Brzy ráno

Po sklíčkách stéká mlhovina,
sporák ohřívá medovinu,
skořice voní v celém domě.

Nad kaplankou svítá.
Nedovřený breviář zívá.
Kříž má kolem sebe stíny.

Pavouci se vrací z nočních vycházek,
Bůh se usmál.

Další den, prázdný skleněně a čiře,
žíznivě čeká na své naplnění.

Venda Veselá

Obsah

Úvod **8**
1.1 Cíl práce 9
1.2 Členění textu 10
1.3 Slovníček 11
1.4 Poděkování 12

Výpočet osvětlení **13**
2.1 Metody typu raytracing 14
2.2 Metody aproximující osvětlení funkcemi 15
2.2.1 Statické scény 15
2.2.2 Dynamické scény 16
2.2.3 Interaktivita v komplexních scénách 17
2.2.4 Určení změnou zasažených linků 17
2.2.5 Aktualizace konfiguračních faktorů 18
2.2.6 Průsečík paprsku se scénou 19
2.3 Discontinuity Meshing 19
2.3.1 Statické scény 20
2.3.2 Dynamické scény 20
2.3.3 Interaktivita v komplexních scénách 21
2.4 Metody založené na rasterizaci 22
2.4.1 Stínová tělesa 22
2.4.2 Promítání 24
2.4.3 Stínové mapy 25
2.4.4 Od difusních odrazů i k lesklým a refrakci 27
2.4.5 Od ostrých k měkkým stínům 27
2.4.6 Od přímého ke globálnímu osvětlení 28

Zrychlení zobrazení a výpočtu **29**
3.1 Kreslení po polygonech, 3D akcelerátor 30
3.2 Práce pouze s viditelnými polygony 30
3.3 Obrazová cache 31
3.4 Stupně detailu 31

Návrh založený na střílení paprsků **33**
4.1 Vnější požadavky 34
4.2 Řešení vyčleněním dynamických objektů 34
4.2.1 Statické objekty 35
4.2.2 Dynamické objekty 35
4.2.3 Vyplývající požadavky 35
4.2.4 Výpočet a zpřesňování statického osvětlení 36
4.2.5 Vliv dynamických objektů na statické osvětlení 36
4.2.6 Přidání statického objektu 40
4.2.7 Odebrání statického objektu 40
4.2.8 Změny dynamických objektů 41
4.3 Adaptivní dělení a interpolace osvětlení 42
4.3.1 Kritérium dělení 43
4.3.2 Eliminace T-vrcholů 44
4.3.3 Interpolace 44
4.4 Clustery 46
4.4.1 Konstrukce 46
4.4.2 Organizace do binárních stromů 46
4.4.3 Vystřelování z clusterů 47
4.4.4 Zásahy do clusterů 47
4.4.5 Shrnutí významu clusterů 48
4.5 Stav implementace 48
4.5.1 Vnější rysy 48
4.5.2 Vnitřní rysy 50
4.6 Shrnutí 51

Návrh založený na stínových mapách **53**
5.1 Indexové stínové mapy v OpenGL 54
5.2 Artefakty indexových stínových map 55
5.2.1 Mizející krajní pixely 55

5.2.2 Mizející polygony 58
5.2.3 Aliasing na hranicích stínů 58
5.2.4 Neosvětlená část prostoru 59
5.3 Výpočet globálního osvětlení 59
5.4 Přesnější řešení 60
5.5 Kompenzace nevystřílené energie 61
5.6 Eliminace duplicitních map 61
5.7 Likvidace vysokých paměťových nároků 62
5.8 Paralelizace 62
5.9 Kreslení jen viditelných polygonů 63
5.10 Cachování v komplexních scénách 63
5.11 Stav implementace 65
5.12 Shrnutí 65

Závěr **66**

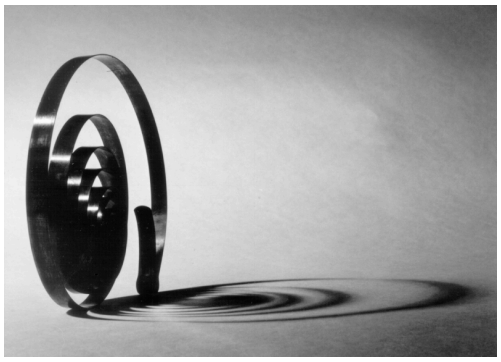
Fotodokumentace **68**

Seznam obrázků

1.1 Ilustrační scéna - „bodáky“. 9
2.1 Ilustrační scéna - „kostky v kostce“. 15
2.2 Ilustrační scéna - „trámy“. 23
2.3 Ilustrační scéna - „disko“. 26
3.1 Ilustrační scéna - „dveře“. 31
4.1 Ilustrační scéna - „kola“. 37
4.2 Ilustrační scéna - „magic view“. 39
4.3 Ilustrační scéna - „jídelna“. 42
4.4 Hrubší dělení zářičů, jemnější dělení příjemců. 44
4.5 Režimy interpolace 45
4.6 Statická scéna - stav po 0.1, 1 a 10 sekundách 48
4.7 Dynamická scéna při 2fps - každý snímek počítán zvlášť 49
4.8 Dynamická scéna při 2fps - aktualizován vliv dynam.objektů . 49
5.1 Ilustrační scéna - „schody“. 54
5.2 Mizející pixely 55
5.3 Ilustrační scéna - „testovací“. 56
5.4 Posun souřadnic v textuře (stínové mapě) 57
5.5 Ilustrační scéna - „zářivka“. 64
6.1 Ilustrační scéna - „zrcadlo“. 67
A.1 Fotodokumentace - studium radiačních metod. 68
A.2 Fotodokumentace - srovnávání s referenčními scénami. 69
A.3 Fotodokumentace - Cornell box. 70
A.4 Fotodokumentace - Zátiší se zrcadlem. 70
A.5 Fotodokumentace - přípitek na další výzkum. 71

1

Úvod



Otáčel v prstech sklenku s vínem a díval se,
jak po bílém stole kloužou červené paprsky.
Nad stolem tiše šuměl plynový hořák.
Umělá palma v rohu pokoje vrhala na stěnu složité stíny.

Jiří Marek, Můj strýc Odysseus

Jedním z nejdůležitějších úkolů při realistické vizualizaci třírozměrné grafiky je rychlý výpočet osvětlení. Správně spočítané stíny, odlesky a další jevy spojené se šířením světla od plošných zdrojů dodávají grafice zdání reálnosti, bez nich je většinou ihned patrné, že jde o syntetický obraz. Stíny a polostíny navíc přispívají k rychlému a správnému vnímání polohy objektů ve scéně a co bývá opomíjeno, hlavní měrou se většinou podílejí na estetické působivosti obrazu.

Výpočtu tzv. globálního osvětlení zahrnujícího veškeré odrazy a cesty světla proto byla věnována velká pozornost a v současné době existují postupy generující velmi kvalitní výsledky. Jejich nevýhodou je ale obvykle značná časová složitost; taková, při které je interaktivní manipulace s geometrií možná jen v jednoduchých scénách. Zatímco v neinteraktivních aplikacích je tedy dnes realistická vizualizace s využitím dlouze předpočítávaných dat standardem, pro interaktivní výpočet osvětlení v dynamicky se měnících komplexních scénách nejsou současné systémy dostatečně výkonné.

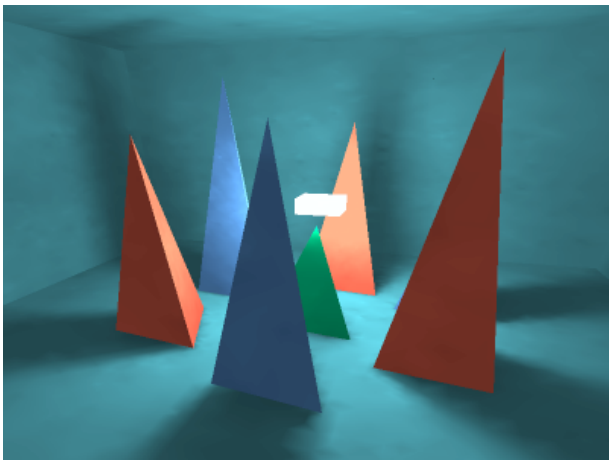
O uplatnění výkonného systému by přitom nebyla nouze. Interaktivní manipulace s geometrií scény je nutná v řadě aplikací, ty jsou přitom dosud odkázány pouze na synteticky vypadající zobrazování bez korektně spočítaného osvětlení. V aplikacích nezbytně vyžadujících správné osvětlení je pak značně omezujícím faktorem velikost dynamicky se měnící scény, u větších scén jsou aktualizace osvětlení velmi pomalé. Pokud je tedy interaktivní výpočet globálního osvětlení dosud omezen jen na jednoduché scény, není nejmenších pochyb, že v budoucnu pronikne i do scén komplexních.

1.1 Cíl práce

Tato práce se zabývá možnostmi vývoje metod výpočtu interaktivního globálního osvětlení směrem k dosud nedosažitelným komplexním scénám; z široké škály známých metod vybírá úzký okruh perspektivních a navrhuje techniky použitelné k jejich dalšímu zrychlení nebo nezbytnému zobecnění. Významnou součástí práce jsou návrhy a částečné implementace dvou vzájemně odlišných přístupů k výpočtu osvětlení.

Oproti původnímu obecnému zadání jsem se tedy ve shodě s vedoucím diplomové práce zaměřil na dosud nezkoumané komplexní scény. Při řešení jsem pak pro úplnost bral v úvahu kromě radiosity i všechny další techniky výpočtu osvětlení. Výstižnější název práce by tak mohl být Interaktivní globální osvětlení v komplexních scénách.

V návrzích a implementaci jsem se orientoval hlavně na dosažení interaktivity a řešení problémů plynoucích z velikosti scény. Související problémy, například jak navrhovaný systém zachycující pouze difusní odrazy rozšířit tak, aby pokryl širší spektrum optických jevů, neřeším, pouze odkazuji na příslušné práce. Při reprezentaci třírozměrné scény jsem se omezil na použití polygonů, rozšiřování k obecnější třídě ploch opět neřeším.



Obrázek 1.1: Ilustrační scéna - „bodáky“.

Považuji zde za vhodné podotknout, že autoři v oblasti osvětlení v dynamických scénách označují při prezentaci vlastních výsledků scény s několika tisíci polygony za komplexní. V tomto směru neexistuje žádná dohoda, přesto považuji za vhodnější a budu slovem ”komplexní” označovat to, co si za ním

většina lidí představí, tedy scény alespoň o řád složitější, řekněme od padesáti tisíc polygonů.

Cílem práce není vysvětlit známé postupy počítačové grafiky a výpočtu globálního osvětlení, ty jsou pouze zařazeny a zhodnoceny po stránce vhodnosti pro interakci v komplexních scénách.

U čtenáře předpokládám znalost základních pojmů a algoritmů počítačové grafiky a OpenGL - ostrý stín, měkký stín (složený z úplného stínu a polostínu), raytracing, radiosita, hierarchická radiosita, Monte Carlo radiosita, discontinuity meshing, wavelet, stínové těleso, stínová mapa, z-buffer, stencil buffer, accumulation buffer apod. Některé z těchto a další pojmy jsou opatřeny odkazy na příslušnou literaturu; jejich znalost není vždy nezbytná, vždy je ale vhodná.

1.2 Členění textu

Následující text je rozčleněn do několika nezávislých kapitol.

Při hodnocení desítek známých algoritmů pro výpočet osvětlení je výhodou jejich schopnost spočítat za delší dobu přesnější výsledek. Nicméně při úvahách o interakci v komplexních scénách je mnohem významnější požadavek dokázat nabídnout nějaký výsledek, byť i horší, rychle. Důležité tedy je, aby čas, za který je k dispozici první nejhrubší výsledek, byl malý a na velikosti scény závisel jen málo, případně vůbec. Při posuzování vhodnosti známých algoritmů i při návrhu vlastních pak bude významným faktorem jejich škálovatelnost, to, jak si poradí s narůstající složitostí scény. U paměťových nároků je téměř předpokladem, že by neměly být výrazně horší, než lineární; v opačném případě by se staly úzkým hrdlem celého systému a záhy zabránily dalšímu zvyšování komplexity scén. Známými algoritmy a jejich vhodností se zabývám v kapitole 2.

Přes všechnu snahu při vývoji algoritmů pro výpočet osvětlení by každý brzy narazil na strop možností současného hardwaru; nebýt ovšem technik, které mohou výpočet a zobrazení značně urychlit, případně umožní i algoritmům tomu nepřizpůsobeným volit svobodně mezi kvalitou a rychlostí. Těmto technikám jsem věnoval kapitolu 3.

Mým cílem, se kterým jsem se pouštěl do oblasti radiačních metod, bylo prakticky demonstrovat, že práce s globálním osvětlením v dynamických scénách je na běžném osobním počítači možná. Pouhý návrh algoritmu bez praktické ukázky bych nepovažoval za uspokojivý. Proto jsem se také neprodleně

pustil do implementace vlastních algoritmů, které jsem v té době považoval za perspektivní. Můj první přístup byl založen na střílení paprsků ze zdrojů světla do celé statické scény (Monte Carlo radiosita) a zvlášť směrem k dynamickým objektům. Pomocí clusterů jsem se snažil zvýšit kvalitu osvětlení v komplexních scénách i při vržení malého množství paprsků. Podrobnější popis řešení založeného na střílení paprsků je v kapitole 4.

Po deseti měsících vývoje metod založených na střílení paprsků, v době, kdy už fungovala práce s dynamickými objekty a průběžné zkvalitňování osvětlení během procházení scénou a kdy se blížil termín odevzdání diplomové práce a zbývalo tedy systém doladit a provést řadu měření, jsem podnikl radikální krok. Ještě ne zcela doladěný systém jsem odložil, zakoupil 3D akcelerátor a vydal se jinou, nyní mnohem slibnější cestou, vycházející ze stínových map. Ta je popsána v kapitole 5.

1.3 Slovníček

Vzhledem k tomu, že většina existující literatury v daném oboru je psána anglicky, připojuji slovníček častěji používaných termínů. Nepřekládané termíny jsou v seznamu ponechány bez překladu, v závorkách jsou uvedeny jiné někdy používané názvy.

aliasing

cluster

depth shadow map, shadow z-buffer hloubková stínová mapa (stínový z-buffer, stínová paměť hloubky)

discontinuity meshing, DM

dynamic discontinuity meshing, DDM

fps, frames per second (snímků za sekundu)

image cache obrazová cache

index shadow map indexová stínová mapa

level of detail, LOD stupeň detailu

lightmap světelná mapa

multi-stage subdivision vícestupňové dělení

shadow map stínová mapa (mapa stínů)

shadow volume stínové těleso

1.4 Poděkování

Během dvanácti měsíců práce mi byl neocenitelnou pomocí Daniel Sýkora (ReDox), který se staral o tvorbu a načítání 3D scén, reprezentaci scén BSP stromem, rasterizaci bez akcelerátoru, vytvořil ilustrační obrázky a animace pro prezentaci zde a na webu, naprogramoval dema používající náš engine a provedl závěrečný layout této práce. Černobílé fotografie v kapitolách 4–7 vytvořil Martin Strejc (Strejda), ostatní černobílé a barevné Pavla Veselá. Všem jsem nesmírně vděčný a děkuji jim za pomoc a podporu.

2

Výpočet osvětlení

Stín v optice vzniká za osvětleným tělesem v prostoru, do něhož světlo nevniká.
Tento stín sluje úplný; vzniká za tělesem, osvětleným bodovým zdrojem světelným anebo zdrojem úhlově malým (velmi vzdáleným). Při větší úhlové velikosti svítících zdrojů vzniká částečný stín, tj. do prostoru za tělesem vnikají paprsky jen z části svítícího zdroje. Částečný stín sluje též polostín.

Teyssler-Kotyška, Technický slovník naučný

Výpočet globálního osvětlení nebo alespoň přesných měkkých stínů od plošných zářičů je oblast intenzivně zkoumaná již mnoho let a existuje pro ni řada různých postupů a technik. Jen malá část z nich se ovšem zabývá interaktivním osvětlením v dynamických scénách a jen ty lepší dosáhly dobrých výsledků alespoň v malých scénách.

V této kapitole zmíním známé postupy výpočtu globálního osvětlení a zaměřím se na ty potenciálně použitelné při interaktivní práci i s komplexními scénami. Na techniky zrychlení výpočtu a zobrazení dojde v další kapitole.

Jelikož třída postupů výpočtu globálního osvětlení je mimořádně rozmanitá a metody se vzájemně prolínají, je třeba vysvětlit podle čeho jsem je rozdělil do následujících sekcí.

Dělení vychází z faktu, že naším cílem je zobrazit povrchy objektů ve scéně nějak osvětlené. Metody lze tedy klasifikovat podle způsobu, jak osvětlení povrchu reprezentují.

V první sekci stručně a pro úplnost zmíním metody typu raytracing. Ty povrchům tvořícím scénu typicky nepřiřazují žádné údaje o jejich osvětlení a vykreslují je po pixelech. V druhé proberu postupy, které povrchům přiřazují jednoduché funkce popisující jejich osvětlení a pokud tato aproximace není

dostatečně přesná, povrchy případně dělí na jemnější elementy. Jako člena této široké třídy metod lze uvést například hierarchickou radiositu s výpočtem konfiguračních faktorů metodou polokrychle. Ve třetí skupině jsou algoritmy, které zjišťují přesné hranice stínů na površích a reprezentují je geometricky. Sem patří tzv. discontinuity meshing. Do poslední skupiny patří metody, které povrchy nedělí, jejich osvětlení reprezentují rastrem, texturou. Takové pak typicky silně využívají 3D akcelerátory.

2.1 Metody typu raytracing

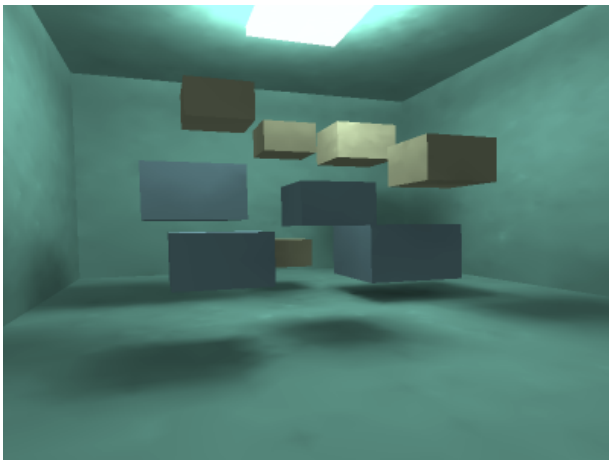
Akceleraci raytracingu a jemu podobných metod založených na sledování paprsků vyslaných z oka pozorovatele, ze světelných zdrojů, případně obojím, byla věnována velká pozornost a dnes je možná interaktivní práce s jednoduššími dynamickými scénami, přičemž zobrazování může zahrnovat řadu optických jevů jinými metodami hůře dosažitelných (odrazy na lesklých plochách, lom světla a to i v závislosti na jeho vlnové délce, průchod světla prostředím). Nový perspektivní přístup, který tyto metody dále zrychluje využitím tzv. "render cache", byl představen v [Wal99]. Pro svůj systém na 195MHz procesoru R10000 a s oknem 256x256 pixelů uvádějí autoři rychlost 14fps (s tím, že jevy jako stín pohybujícího se objektu se počítají průběžně, stín postupně mizí ze staré pozice a objevuje se v nové) a očekávají další zrychlení, například s využitím SIMD instrukcí. Mimo to lze výpočet dobře paralelizovat a provádět na více procesorech. Ve velmi komplexních scénách, kde se do záběru dostává jen malá část geometrie scény a ovlivňuje ho jen malá část světelných zdrojů (místnost v budově apod.) navíc raytracing přirozeně eliminuje nebo alespoň redukuje náklady na to, co pro současný záběr nemá význam, co je skryté. Věřím, že tento přístup má budoucnost a bude se dále vyvíjet. Nicméně nevýhody všech podobných metod stále zůstávají, náročnost zhruba lineárně roste s počtem zobrazovaných pixelů, při rychlém natočení kamery jiným směrem je nutné prakticky vše přepočítat, resp. dočasně značně klesne kvalita obrazu. Pokud si můžeme dovolit pokročilé optické jevy omezit a spokojíme se s jednoduššího osvětlovacím modelem, pro interaktivní práci s komplexními dynamickými scénami tedy tyto metody nepovažuji za nejvhodnější a dále se jimi nebudu zabývat.

Totéž platí pro hybridní metody ukládající pomocné informace do povrchů, ale stále založené na pomalém vrhání paprsků a vykreslování scény po pixelech.

2.2 Metody aproximující osvětlení funkcemi

Pod tímto názvem se skrývá většina klasických (nejstarších a nejvíce vyučovaných) technik z oblasti radiosity. Osvětlení povrchů aproximují nějakou funkcí, nejčastěji konstantou, a tam, kde tato aproximace nestačí, obvykle povrch dělí na menší elementy a to bez snahy dělit přesně podle hranice stínu.

Před vlastní úvahou nad nejvhodnějším přístupem k zobrazování komplexních dynamických scén s globálním osvětlením v sekci 2.2.3 bude vhodné zmínit dosud publikované postupy pro dynamické i statické scény.



Obrázek 2.1: Ilustrační scéna - „kostky v kostce“.

2.2.1 Statické scény

Od počátečního "full-matrix" přístupu [Gor84] nad pečlivě nameshovanou scénou prošla radiosita ve statických scénách dlouhým vývojem, objevilo se adaptivní dělení povrchu (adaptive subdivision) [Coh86] odstraňující potřebu polygony scény dělit předem a často zbytečně, rychleji konvergující progresivní řešení (progressive refinement, postupující radiosita) [Coh88], hierarchická radiosita [Han90] zefektivňující propojení ploch linky, aproximace osvětlení povrchu funkcemi vyšších řádů [Zat93], směrový přístup umožňující použít obecnější třídu materiálů, např. [Hal93], waveletová radiosita [Gor93], přesný analytický výpočet faktoru polygon-polygon [Sch93], clustery rozšiřující hierarchii v hierarchické radiositě [Sil94, Smi94] (přehled v [Has99]),

efektivnější metriky pro rozhodování které povrchy nebo linky zjemnit, např. [Sil95], visibility skeleton umožňující ještě rychlejší výpočet viditelnosti [Dur97], přizpůsobení clusterů pro velmi komplexní scény ve face-cluster radiosity [Wil99] a konečně i velmi rychlé Monte Carlo metody střílející všesměrově tolik paprsků, kolik je třeba vyzářit energie (porovnání v [Bek99]), spojené s rychlejší konvergenci přinášejícím Quasi Monte Carlo vzorkováním [Kel96] a hierarchickým dělením povrchů [Tob97, Bek98],

2.2.2 Dynamické scény

Jak je to ale se známými postupy pro dynamické scény? První z prací pokrývaly pouze speciální případy, například pohyb po předem známé dráze [Bau86]. Obecný případ změny globálního osvětlení následkem pohybu dynamických objektů pak řešili [Geo90, Che90]. Jejich přístup k aktualizaci byl eliminovat vliv dynamického objektu vystřílením záporné energie, umístit ho na novou pozici a střílením kladné energie započítat jeho vliv v nové poloze. Další práce se zabývaly zvyšováním efektivity tohoto postupu [Mul94] a použitím v hierarchické radiositě [For94, Sha94]. K rozeznání které linky pohyb objektu ovlivnil byl kolem staré a nové pozice objektu konstruován tzv. motion volume, objem, ve kterém došlo k pohybu.

Tyto postupy používaly adaptivní dělení povrchů, konstantní intenzitu osvětlení či vyzařování ukládaly pouze v polygonech nebo ve vrcholech a tam, kde by tato aproximace zavinila příliš velkou chybu, dělily původní polygon na jemnější elementy.

Další práce šly směrem lepší detekce měnících se linků. Line-space hierarchy [Dre97] říká které linky je třeba po změně geometrie aktualizovat a umožňuje i snadné odstranění takového zjemnění povrchu, které po změně osvětlení není potřeba (např. zjemnění na bývalé hranici stínu). Přitom lze plynule volit mezi rychlostí a přesností výsledku. Na 200MHz Indigo2 R4400 uvádějí autoři rychlost 2fps (frames per second, snímků za sekundu) ve scéně s 870 polygony a 0.3fps ve scéně s 5295 polygony. Dynamické stíny jsou přitom výrazně zubaté. Kvalita obrazu s interpolací by byla vyšší, ovšem rychlost celé metody zřejmě silně závisí na počtu významných zářičů a výsledky by byly ve scénách s více než jedním zářičem horší.

2.2.3 Interaktivita v komplexních scénách

A jak je to s vhodností všech výše uvedených přístupů pro komplexní dynamické scény?

Některé metody můžeme ihned vyloučit, např. waveletová radiosita je ve srovnání s ostatními příliš paměťově náročná.

Visibility skeleton, jak byl prezentován v [Dur97], je při zodpovídání dotazů na viditelnost velmi efektivní, ale ve scéně s 1500 polygony spotřebuje 240MB paměti a jeho konstrukce trvá půl hodiny, efektivní aktualizace pro dynamické scény jeho tvůrci uvažovali do budoucna. V [Dur99] visibility skeleton dále rozšířili a mírně snížili paměťové nároky, dynamické scény ovšem nadále uvádějí jako výhled do budoucna.

Face-cluster radiosity [Wil99] umožňuje díky hierarchii clusterů rychlý výpočet i ve velmi komplexních scénách. Budování hierarchie pro statickou scénu ovšem trvá řádově déle než následný výpočet osvětlení. Dobré výsledky jsou dosaženy zčásti proto, že většina geometrie scény není při výpočtu vůbec použita, místo ní ve výpočtu figurují pouze hrubší clustery aproximující jemnější geometrii. Navržené clustery z face-cluster radiosity byly použity pouze k výpočtu, nicméně nic nebrání tomu, aby se uplatnily i při zobrazování. Face-cluster radiosity se pak může po kombinaci se strukturami určujícími které linky je třeba aktualizovat stát základem pro systém umožňující interakci i ve velmi komplexních dynamických scénách.

Využití různých úrovní detailů jsem ovšem zařadil mezi metody zrychlující výpočet a dále se zabývám pouze samotným výpočtem nad jednou úrovní detailů a jejím zobrazením bez zjednodušování geometrie.

Jako nejvhodnější metoda k dosažení interaktivity v komplexních scénách se jeví hierarchická Monte Carlo radiosita s clustery a adaptivním dělením povrchů. Toto jsem implementoval a popsal v kapitole 4.

2.2.4 Určení změnou zasažených linků

K dosažení dobrých výsledků je ale nutno vyřešit nejprve několik problémů. Jedním z nich je jak určit co po změně geometrie aktualizovat. Odpověď na tuto otázku dává například line-space hierarchy [Dre97]. Ve své implementaci jsem se určování zasažených linků vyhnul, použil jsem jednodušší střílení paprsků od nejvýznamnějších zářičů směrem k pro ně nejvýznamnějším dynamickým objektům a z dynamických zářičů do scény, jak bude popsáno v kapitole 4.

2.2.5 Aktualizace konfiguračních faktorů

Jiný významný problém je jak změnou geometrie ovlivněné údaje aktualizovat, konkrétně většinou jak aktualizovat nějaký typ konfiguračních faktorů, obvykle polygon-polygon nebo vrchol-polygon.

Pokud chceme spočítat jeden faktor, existuje řada řešení. Pro případ ničím nezastíněné dvojice polygonů Monte Carlo integrace [Coh93] nebo přesné analytické řešení [Sch93], pak je ovšem nutné do výsledku ještě nějak zahrnout viditelnost. Tu lze určit buď geometricky ořezáním všech stínících polygonů a spočítáním nezastíněné části [Nis85] nebo vrháním paprsků z jednoho polygonu směrem k druhému a počítáním jaká část jich volně projde [Han91]. Pro komplexní dynamické scény jsou ale všechny tyto metody, vzhledem k tomu, že počítají pouze jeden faktor, poměrně pomalé.

Mimo výpočet jediného faktoru je často potřeba spočítat všechny faktory z jednoho polygonu. Pak existují podstatně efektivnější metody než postupné počítání faktorů ke všem ostatním polygonům ve scéně.

První je střílení paprsků z výchozího polygonu do prostoru a evidování zasažených okolních polygonů. Pokud budeme směry střílených paprsků volit tak, aby hustota jejich pravděpodobností odpovídala hustotě energie opouštějící zářič, případně paprskům přiřadíme vhodné váhy, počty zásahů budou odpovídat množství energie dopadající na okolní polygony. Nevýhodou je fakt, že drobné nerovnoměrnosti v distribuci paprsků různými směry vnesou do výsledku šum. Výhodou je plná kontrola nad rychlostí výpočtu - můžeme střílet jak dlouho budeme chtít a výsledek bude stále přesnější. Naopak lze vystřelit nepatrné množství paprsků a jakýsi výsledek, ač nepřesný, také obdržíme a to prakticky ihned.

Druhý přístup, nazývaný podle varianty metoda polokrychle, jedné průmětny apod., má při dané scéně časovou složitost zhruba konstantní a to ve srovnání s vystřelením pár paprsků relativně vysokou, výsledky ovšem dává velmi přesné a nezatížené takovým šumem. Scéna je vykreslena z pohledu z výchozího polygonu, každý polygon scény jinou barvou. Každý pixel obrazu má svou váhu odpovídající množství energie vycházející z bodu na polygonu a procházející plochou pixelu. Faktor z výchozího bodu k polygonu pak získáme jako součet vah pixelů barvy, jakou byl tento polygon vykreslen.

Název metoda polokrychle znamená, že scéna je vykreslena do pěti stěn polokrychle obklopující bod na výchozím polygonu. Používaná alternativa je vykreslit scénu jen do jedné průmětny, do jednoho čtvercového rastru nad výchozím bodem. Její nevýhodou je, že čtverec nad výchozím bodem rovnoběžný s výchozím polygonem nepokrývá všechny směry, kterými světlo

z bodu vychází. Proto se někdy k prvnímu rastru přidává druhý, s většími pixely, ale o to širší, tím se pokrytí zvyšuje. Všechny paprsky vylétající z bodu do poloprostoru lze plně pokrýt třemi čtvercovými rastry svírajícími pravé úhly. Toto řešení jsem neviděl nikde použité ani popsané, ale domnívám se, že při práci s 3D akcelerátorem je vhodnější než polokrychle. Čím více je vykreslování (jedno, dvě, tři nebo pět), tím je operace na 3D akcelerátoru pomalejší, neakcelerované kreslení je pomalé vždy. Pokud záleží hlavně na rychlosti, jeden rastr pro zachycení nejdůležitější části osvětlení stačí.

V našem případě (komplexní dynamické scény) přichází v úvahu rychlejší střílení paprsků i kvalitnější jeden rastr, záleží na situaci. Ve své implementaci popsané v kapitole 4 jsem se zaměřoval pouze na rychlost a spokojil se se střílením paprsků.

2.2.6 Průsečík paprsku se scénou

Vzhledem k tomu, že výpočet konfiguračních faktorů bývá při výpočtech globálního osvětlení časově nejnáročnější operací a počítání průsečíků paprsků se scénou může být časově nejnáročnější částí výpočtu konfiguračních údajů, je třeba věnovat mu velkou pozornost. Počítání průsečíků se uplatňuje i v raytracingu a i tam hraje klíčovou roli, takže je jisté, že možnosti jeho urychlení byly důkladně prozkoumány. Mezi datové struktury používané pro reprezentaci scény patří různé varianty BSP stromů [Fuch80], octree [Gla84], pravidelných [Fuj86] i adaptivních mřížek, KD stromů, hierarchií obalových těles [Kay86] a další. Pro všechny takové struktury existují algoritmy jejich průchodu a nalezení případného průsečíku. Rychlost a další vlastnosti těchto struktur v současné době porovnává projekt GOLEM (http://www.cgg.cvut.cz/GOLEM). V rámci naší spolupráce s Danielem Sýkorou připadla reprezentace geometrie scény a hledání průsečíku na něj. Z uvedených struktur zvolil BSP stromy.

2.3 Discontinuity Meshing

Metody popsané v předchozí sekci trpí tím, že se dělení ploch na menší elementy řídí pouze tvarem těchto ploch, nikoliv tvarem stínů. Pro dostatečně přesnou aproximaci tvaru stínů pak vyžadují velmi jemné dělení v částečně zastíněných oblastech a s ním roste i časová náročnost. Počet elementů i dobu výpočtu by přitom velmi výrazně snížilo rozdělení ploch na menší elementy

přesně na hranicích stínů a diskontinuit, které uvnitř stínů vznikají. Takové dělení se nazývá discontinuity meshing (DM), jeho adaptace pro dynamické scény dynamic discontinuity meshing (DDM).

2.3.1 Statické scény

První práce počítající hranice stínů (ovšem ještě ne diskontinuity uvnitř polostínů) [Nis83] k tomu používala stínová tělesa (viz. sekce 2.4.1). Výsledky tehdy ještě nebyly uplatněny v oblasti radiosity. Zrychlení výpočtu s použitím BSP stromů při reprezentaci stínových těles popsali [Cam91, Chi92], [Cam91] přitom aproximuje nerovnosti v intenzitách polostínů přesným výpočtem osvětlení v navzorkovaných bodech a interpolací mezi nimi.

Discontinuity meshing jako metoda vhodného dělení ploch pro radiositu byla uvedena nejprve pro dvourozměrný případ [Hec92a], později v zobecnění pro 3D nezávisle v [Hec92b, Lis92]. Druhá práce byla posléze zkombinována s hierarchickou radiositou [Lis93].

Všechny algoritmy ale pracovaly jen s částí diskontinuit. Diskontinuity ve scéně z polygonů mohou obecně nabývat i tvaru křivek a zahrnout do řešení i ty je obtížnější, kompletní řešení tak ukázali až [Dre94, Ste94]. Nicméně v praxi se obvykle pracuje stejně jen s částí diskontinuit, méně významné a obtížně spočitatelné jsou z řešení pro zvýšení rychlosti vypuštěny.

Další vývoj ve statických scénách přinesl algoritmy a datové struktury akcelerující výpočet DM. [Ort96, Dur97] představují již dříve zmiňovaný visibility skeleton, víceúčelovou strukturu udržující informace o viditelnosti ve scéně a umožňující mimo jiné i velmi rychlý výpočet DM. Konstrukce samotného visibility skeletonu je ovšem časově i prostorově velice náročná a problém jeho dynamických aktualizací zatím nebyl vyřešen.

2.3.2 Dynamické scény

Při práci v dynamických scénách je potřeba uvážit, že spolu se změnami geometrie objektů se mění i hranice stínů. Pokud použijeme starší statické metody a pro každý snímek znovu spočítáme hranice stínů, výsledný program bude příliš pomalý. Visibility skeleton by dokázal hranice stínů počítat dostatečně rychle, ale sám je konstruován jen pro statickou scénu a konstrukce nového skeletonu pro nový snímek nepřichází v úvahu.

Žádoucí by bylo buď

- efektivně rozhodnout, které údaje změna scény ovlivní a pak znovu spočítat pouze ty. To je postup použitelný i pro metody používající adaptivní dělení povrchů a práce jdoucí tímto směrem už byly zmíněny v 2.2.2.

- nebo vyvinout a udržovat ve scéně datové struktury umožňující hranice stínů ne přepočítávat, ale efektivně aktualizovat. Tento přístup se nazývá dynamic discontinuity meshing.

První práce jdoucí směrem DDM je [Wor95]. Autor svou metodu dále vyvíjel [Wor98], pro výpočet primárního osvětlení v dynamické scéně s 8000 polygony uvádí rychlost 1fps na 167MHz UltraSparc. Podle srovnání rychlostí v různě velkých scénách je vidět, že jeho algoritmus jen málo závisí na složitosti scény, jako u všech ostatních DDM řešení ovšem rychlost závisí na množství změn ve scéně a měření jsou obvykle prováděna při pomalém pohybu jediného objektu.

Další algoritmy aktualizující discontinuity meshing ve scéně přinesli [Los97] (při výpočtu primárního osvětlení ve scéně s 500 polygony uvádí rychlost 0.4fps na 150MHz Indigo2 R4400) a [Chr97] (ve scénách obsahujících 92-184 polygonů se rychlost pohybuje v rozmezí 0.4-1.2fps na 75MHz SUN SparcStation 20).

2.3.3 Interaktivita v komplexních scénách

Oblast DDM se vyvíjí teprve pět let a vzhledem k obtížnosti problémů, které nastoluje, se jí vážněji zabývá jen poměrně malé množství lidí. Dosud publikované práce umožňují interaktivitu zatím jen v malých scénách, přesto považuji DDM za nadějný směr, např. výsledky [Wor98] ukazují, jak málo může rychlost aktualizace osvětlení záviset na složitosti scény. Ve scénách s rapidně se měnící geometrií jsou dosud DDM řešení příliš pomalá, ovšem pokud jde o změny osvětlení způsobené drobnými změnami geometrie, zůstane DDM pravděpodobně nejrychlejší cestou, jak dosáhnout velmi kvalitních výsledků. Jedinou nevýhodou pro použití ve velmi rozsáhlých scénách se může stát vyšší paměťová náročnost. V takových situacích bude nutné zřeknout se vysoké kvality a použít některou z méně náročných metod využívajících 3D akcelerátory.

2.4 Metody založené na rasterizaci

Poslední třída metod výpočtu osvětlení zahrnuje ty, které reprezentují viditelnost nebo osvětlení polygonu rastrem (viditelnost tzv. stínovou mapou, osvětlení tzv. světelnou mapou), případně vytváří stíny přímo v rastru, do kterého scénu vykreslují. Polygony tvořící scénu díky tomu nemusí dělit na jemnější elementy a přesto dokáží zachytit jemné detaily. Mapy ovšem mají nějaké konečné rozlišení limitované množstvím paměti nebo schopnostmi 3D akcelerátoru, takže detaily zachycují jen do určité úrovně, za ní mohou podle konkrétní metody vznikat různé artefakty, například aliasing na hranicích stínů.

Společnou vlastností všech rastrových metod je to, že počítají pouze primární osvětlení a často jen z bodových zdrojů. Rozšíření od bodových k aproximaci plošných zdrojů jsou ovšem známa a efektivní rozšíření k aproximaci globálního osvětlení navrhnu.

Nevýhodou všech předchozích metod, které se snaží aktualizovat pouze ty údaje, na které má změna ve scéně vliv, je to, že pokud jsou změny příliš rozsáhlé, aktualizace se v podstatě neliší od kompletního přepočítání osvětlení a to bývá stále časově velmi náročné, neúnosné pro interakci.

Naproti tomu metody založené na rasterizaci bývají díky využití 3D akcelerátorů natolik rychlé, že i při úplném přepočítání osvětlení je možná interakce. Využití informací z dřívějších snímků pak není nutné, ovšem v případě, že k němu přikročíme a nebude nutno přepočítat vše, budeme moci pracovat s výrazně složitějšími scénami.

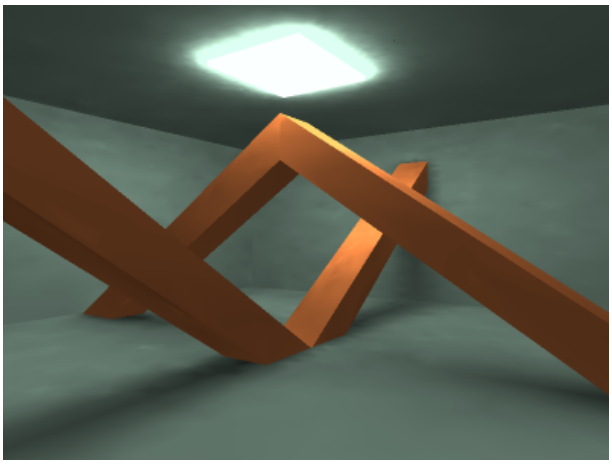
Nejprve se tedy podíváme na akcelerovatelné metody výpočtu stínu z jednoho bodového zdroje světla, poté přijde zobecňování k měkkým stínům, globálnímu osvětlení a komplexním scénám.

Rastrovými metodami nevhodnými pro efektivní akcelerovanou a tedy rychlou implementaci, např. řádkovým algoritmem, algoritmem dělení [Ath78] nebo systémem prezentovaným v [Tan97], se zabývat nebudu.

2.4.1 Stínová tělesa

Roku 1977 byl představen způsob jak počítat ostré stíny z bodového zdroje světla pomocí tzv. stínových těles (shadow volumes) [Cro77]. Pro každý objekt, který může vrhat stín, jsou zkonstruovány plochy ohraničující zastíněný prostor (stínové těleso), plochy ubíhající od objektu ve směru paprsku do nekonečna. Po vykreslení scény pak lze vhodným porovnáním vzdáleností těchto

ploch a vzdáleností uložených v z-bufferu rozlišit pixely ležící ve stínovém tělese a pixely ležící mimo něj. Zvlášť důležité je, že všechny potřebné operace může provádět standardní 3D akcelerátor vybavený stencil bufferem, jak ukazují [Fuch85, Hei91], díky tomu lze obrazy se stíny generovat velice rychle. [Ude99] dále ošetřili krajní případy, ke kterým při práci se stínovými tělesy dochází, [Die94] metodu rozšířili o efektivní simulaci zrcadlového odrazu stínu a lomu stínu při průchodu do jiného materiálu (obdoba odrazu a lomu světla). Počítání ploch ohraničujících stínová tělesa ovšem klade značné nároky na procesor, pro jednoduché scény zanedbatelné, ovšem pro komplexní už značné. Určitou nadějí pro komplexní scény je nový přístup detekující hranice stínových těles analýzou diskontinuit v z-bufferu po vykreslení z pohledu ze světla. V obou případech je ale značně zatížen procesor a sběrnice mezi ním a akcelerátorem. V dnešní době extrémně rychle rasterizujících akcelerátorů se přitom úzkým hrdlem na architektuře PC stává právě sběrnice a procesor, nutnost generovat dodatečné polygony na procesoru a přenášet je do karty je nevýhodou objemových stínů. Ta je ovšem vyvážena vysokou přesností při určování stínů; díky přímé rasterizaci ve výsledném obraze nejsou stíny zatíženy běžnými artefakty vznikajícími při mapování stínů z jednoho rastru do druhého (tj. ze světelných nebo stínových map do výsledného obrazu).



Obrázek 2.2: Ilustrační scéna - „trámy“.

Zvláštní vlastností metody stínových těles je, že generuje stíny pouze pro daný snímek, informace o stínech mimo záběr nelze získat ani jako vedlejší produkt. Metoda je tedy nevhodná pro snadné rozšiřování ke globálnímu osvětlení, vhodnější budou postupy počítající veškeré přímé osvětlení i mimo současný záběr. Stínová tělesa pak lze pro jejich přesnost doporučit tam, kde jde pouze o výpočet primárního osvětlení.

2.4.2 Promítání

V době, kdy bylo úspěchem zobrazovat stín objektu na jedné ploše, vznikla jednoduchá metoda - projekcí objektu z bodového zdroje světla na vzdálenější plochu vznikne silueta objektu odpovídající jeho stínu [Bli88]. Promítání objektů na plochu lze provádět přesně geometricky, rychlejší je ale rasterizace objektů z pohledu ze zdroje světla. Výsledný rastr můžeme uložit jako tzv. světelnou mapu (lightmap) a to buď jednobitovou, rozlišující pouze pixely osvětlené a zastíněné, nebo vícebitovou, udávající u osvětlených bodů i intenzitu osvětlení. Při pozdějším zobrazování ploch na ně aplikujeme jejich světelné mapy.

Při generování textur i texturování se zapnutou bilineární interpolací se zbavíme rušivě působících naprosto ostrých a někdy viditelně zubatých hranic stínu. Čím nižší rozlišení pro texturu použijeme, tím širší okraj stínu bude tvořit plynulý přechod mezi stínem a osvětlenou plochou, vhodnou volbou rozlišení pak můžeme vyvolat zdání měkkých stínů. Přitom ale půjde jen o vedlejší produkt texturování, který nemá s korektně spočítanými měkkými stíny nic společného.

V naší situaci, kdy chceme zobrazit stíny na všech plochách tvořících scénu, by bylo nutné generovat světelné mapy pro všechny polygony ve scéně. V případě komplexních scén je to neúnosné už jen pro paměťovou náročnost, i při menším počtu polygonů je ale patrné, že jde též o časově náročnou operaci - vygenerovat jednu světelnou mapu znamená promítnout na plochu všechny potenciálně stínící objekty. Těch může být hodně, generování všech map je tedy značně pomalé.

Počet generovaných map můžeme snížit tak, že nebudeme generovat jednu pro každý polygon, ale jednu pro každý objekt a poté ji na něj promítneme. Promítat ale můžeme pouze na konvexní objekty nebo konvexní komponenty, promítáním světelné mapy na nekonvexní objekt by u něj nemohlo vzniknout sebezastínění. Zásadní nevýhoda této optimalizace ale je, že neumožňuje skládání více světelných map od různých zdrojů do jedné mapy, a tedy ani efektivní rozšíření k plošným zářičům.

Je možné i zcela vypustit někdy paměťově neúnosné generování světelných map a geometrii scény promítat na plochy přímo při jejich zobrazování. Pak ale přijdeme o vítanou možnost aktualizovat v každém snímku jen vhodnou malou část světelných map, ostatní nechat za cenu malých nepřesností neaktuální a tím procházení scénou podstatně zrychlit.

Promítání geometrie scény na každý polygon nebo objekt je nejvhodnější v situaci, kdy očekáváme procházení statickou scénou, kdy světelné mapy

stačí spočítat jen jednou a poté je používat beze změn. To ale není náš úkol, tím jsou dynamické scény. Překvapivě se objevily práce používající světelné mapy i v dynamických scénách [Hec97b, Her97]. V takovém prostředí se nabízí řada optimalizací - aktualizovat v každém snímku jen vhodnou podmnožinu světelných map, při generování map využít struktury udržující v dynamické scéně seznamy potenciálně stínících objektů mezi zářičem a plochou a kreslit jen ty, při softwarové implementaci kreslit stínící objekty do tzv. span bufferů [Her97]. Přesto ale zůstává vysoká časová složitost a pro komplexní scény to není vhodný přístup. Nebudu se tedy ani zabývat rozšiřováním promítání ke globálnímu osvětlení.

2.4.3 Stínové mapy

Další již dlouho známou technikou generování stínů jsou stínové mapy (mapy stínů, shadow maps) [Wil78]. Viditelnost počítají tak, že nejprve vykreslí scénu z pohledu zářiče a výsledek uloží do tzv. stínové mapy. V ní je vidět právě to, co zářič osvětluje (jako vedlejší efekt tedy získáme přehled o všech přímo osvětlených plochách, ne jen o osvětlení zobrazovaných ploch; to se bude hodit při rozšiřování ke globálnímu osvětlení). Při kreslení scény z pohledu kamery pak stačí pro každý kreslený pixel otestovat, jestli je vidět ve stínové mapě a je tedy osvětlený. Takový test vyžaduje transformovat každý pixel z prostoru kamery do prostoru zářiče, což by bylo značně časově náročné, [Seg92] ovšem ukázali, jak k transformaci využít texturovací hardware.

Samotné testování může mít dvě podoby – být založené na porovnávání hloubky nebo indexu.

Hloubkové stínové mapy

První varianta algoritmu stínových map pracující s tzv. hloubkovou stínovou mapou (depth shadow map, shadow z-buffer, stínový z-buffer, stínová paměť hloubky) porovnává souřadnici z získanou transformací pixelu do prostoru zářiče a z uložené na tomtéž místě v hloubkové stínové mapě. Pokud se shodují, pixel je ze zářiče viditelný a tedy osvětlený. Při takovém testování je ovšem nutné vyrovnat se s numerickými nepřesnostmi způsobujícími sebezastínění nebo chybějící stíny [Woo92]. [Ree87] představili způsob jak antialiasovat hranice stínů, ovšem pouze softwarově. Provádět testování hardwarově, jak předvedli [Seg92], vyžaduje akcelerátor podporující textury obsahující složku z, údaj o hloubce pixelu, což není běžné a není standardní

součástí OpenGL (jde o volitelně podporovaná rozšíření SGIX_depth_texture a SGIX_shadow).

Jak hardwarově implementovat stínové mapy i s běžným OpenGL údajně (podle [Hei00]) ukazuje [Bra00], tato práce ovšem není dostupná a nelze zjistit detaily, např. zda nejde o indexové stínové mapy.

Indexové stínové mapy

Druhou možností jsou tzv. indexové stínové mapy. Vzniknou tak, že každý polygon vykreslíme unikátní barvou, indexem. Testování se pak redukuje na prosté porovnání indexu kresleného polygonu a hodnoty v indexové stínové mapě.

Stejně jako u hloubkových stínových map jsou i všechny mně známé hardware využívající implementace indexových stínových map založeny na proprietárních rozšířeních OpenGL. Je tedy možné, že moje řešení popsané v sekci 5.1 jako první ukazuje jak implementovat indexové stínové mapy s OpenGL bez rozšíření.

Obrázek 2.3: Ilustrační scéna - „disko“.

Indexové stínové mapy trpí podobně jako hloubkové mapy řadou artefaktů. Zatímco ale u hloubkových map byly většinou na vině numerické chyby při výpočtu z, zde je většinou problémem konečné rozlišení map a jím způsobený aliasing. Způsoby ošetření těchto chyb navrhnu v 5.2.

2.4.4 Od difusních odrazů i k lesklým a refrakci

Zobecňování předchozích metod tak, aby zachytily co největší škálu optických jevů, byla věnována značná pozornost a existuje mnoho různých postupů jak toho částečně nebo zcela dosáhnout při menším nebo větším zpomalení.

Abych dále nerozšiřoval již tak široký záběr této práce a mohl se soustředit na jiné oblasti, omezil jsem se při svém dosavadním experimentování s rastrovými metodami na jednoduchý model řady difusních odrazů a v závěru jednoho lesklého odrazu pouze od zdrojů světla (podrobněji v kapitole 5) a dalším rozšiřováním se nezabýval.

Pro případné zájemce přesto z řady prací vybírám [Die94] nebo podrobnější [Die96] popisující jak v kombinaci s objemovými stíny na akcelerátoru efektivně dosáhnout zrcadlových odrazů a refrakce. Přehled o dalších technikách lze získat například z [Hei99].

2.4.5 Od ostrých k měkkým stínům

Známe-li způsob, jak efektivně zobrazit ostré stíny z bodového zdroje světla, přímočarý a nejběžnější způsob generování měkkých stínů od plošného zdroje světla je aproximace zdroje vícero body a složení, zprůměrování vzniklých ostrých stínů.

První práce z osmdesátých let [Bro84, Ber86] využívaly např. rozšířený z-buffer se spojovým seznamem pro každý pixel a nebyly tedy akcelerovatelné. Dnešní standardní způsob jak stíny složit je složit celé výsledné obrazy scény v accumulation bufferu [Hae90], s lepšími nebo novějšími akcelerátory tedy velmi efektivně. Accumulation buffer je velmi univerzální a použitelný prakticky u všech rastrových metod.

Pro akcelerátory bez rychlého hardwarového accumulation bufferu jsem v případě stínových map implementoval ještě alternativní skládání pomocí běžněji hardwarově podporovaného blendingu, snímky se generují tmavší a místo průměrování se sečtou. Sčítání probíhá na úrovni pixelů už při vykreslování polygonů, odpadá tak následné sčítání na úrovni celého bufferu. Nicméně s narůstajícím počtem skládaných obrazů roste vliv zaokrouhlovacích chyb a klesá kvalita výsledku. Blending lze použít pouze ke snížení počtu pomalých operací nad accumulation bufferem (například blendovat dohromady každých osm snímků a teprve tento mezivýsledek vložit do accumulation bufferu).

Odlišný přístup ke skládání ostrých stínů prezentovali [Hei00]. Pro případ lineárních světel (svítící úsečky) generují jen dvě okrajové stínové mapy a

jejich analýzou určí všechny oblasti polostínu. V těch poté osvětlení vhodně interpolují. Jde o pouhou aproximaci skutečného osvětlení, ve složitějších scénách vznikají artefakty, nicméně dojem z interpolovaných polostínů je vždy mnohem lepší, než z hrubých polostínů obsahujících jen pár úrovní. Autoři navrhli i adaptaci pro určitou aproximaci osvětlení plošným zdrojem. Analýza stínových map je ovšem časově náročná, proto jsem při vlastním vývoji před interpolací upřednostnil klasické skládání většího počtu ostrých stínů, při kterém se lze navíc volněji rozhodovat mezi rychlostí a kvalitou.

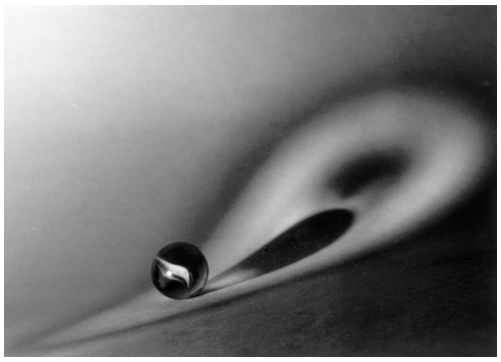
2.4.6 Od přímého ke globálnímu osvětlení

Dosud jsme se v rastrových metodách zabývali pouze přímým osvětlením od primárních zářičů.

Práci předvádějící účinné rozšíření některé z rastrových metod ke globálnímu osvětlení jsem nenašel žádnou, proto jsem v kapitole 5 navrhl vlastní postup založený na stínových mapách (pro objemové stíny tento postup nelze použít, pro metody založené na promítání by adaptovat šel, ale ty jsem vyloučil jako neefektivní). Součástí návrhu budou i optimalizace urychlující práci s komplexními scénami.

3

Zrychlení zobrazení a výpočtu



... potom se uzavřel ve stáji, načež zhasla všechna světla,
- pro koho by se mělo svítit? - a jenom nahoře v dřevěné galérii
zůstala jasná štěrbina a trochu poutala bloudící zrak.

Franz Kafka, Zámek

Tato kapitola je věnována různým metodám, jejichž použití může zrychlit výpočet osvětlení i zobrazení scény a které přitom často nejsou na způsobu samotného výpočtu osvětlení závislé a lze je pak případně i kombinovat, vzájemně se nevylučují.

Vzhledem k tomu, že práce zabývající se výpočtem osvětlení se dosud jen řídce zaměřovaly na komplexní scény, nebyla ani velká potřeba zabývat se metodami urychlení výpočtu a zobrazení, které v malých scénách nepřinášejí výrazný užitek. Nicméně pouhým zobrazováním značně rozsáhlých scén (bez výpočtu osvětlení) se už zabývala řada prací a jsou známy velmi efektivní postupy. Díky tomu, že rychlost některých z nejefektivnějších metod výpočtu osvětlení (stínové mapy, viz sekce 2.4.3) je přímo úměrná rychlosti zobrazení scény, metody urychlení zobrazení lze s menšími či většími úpravami použít i k urychlení výpočtu osvětlení.

Jelikož většina metod urychlení, které zde zmíním, je implementačně náročná, sám jsem se jimi většinou nezabýval, uvádím je jen jako možnosti pro další vývoj. V hypotetickém finálním produktu našeho vývoje, v nejlepším možném vizualizačním enginu, by obsaženy být měly, jejich přínos je nenahraditelný.

3.1 Kreslení po polygonech, 3D akcelerátor

Prvním a prakticky nezbytným krokem k interaktivitě v komplexních scénách je vykreslení polygonové scény po celých polygonech místo po jednotlivých pixelech. K vykreslení polygonů lze navíc využít grafické akcelerátory, k vykreslení celé scény pak stačí zlomek času, který by spotřeboval softwarový rendering polygonů nebo raytracing.

Akcelerované zobrazení nezmění kvalitu v případě zcela matných plošek. Situace v oblasti lesklých odrazů, lomu světla a dalších optických jevů je složitější.

Jednoduché a přímočaré řešení je v kombinaci s raytracingem. Scéna se nejprve vykreslí po ploškách s pouze matným odrazem, přičemž ty, na kterých by tím vznikala příliš velká chyba, dostanou speciální barvu, při druhém průchodu se všechny pixely speciální barvy přepočítají raytracingem. V reálných scénách s množstvím lesklých ploch to ovšem zobrazování značně zpomalí.

Naštěstí lze lesklé odrazy a lom světel i stínů relativně efektivně aproximovat i bez vrhání paprsků, pouze využitím akcelerátoru, jak ukazuje [Die96].

Uvedené metody lze alespoň k zobrazování použít prakticky s jakýmkoliv algoritmem na výpočet globálního osvětlení (výjimkou jsou metody založené na raytracingu). Největší efekt ovšem přinesou v rastrových metodách (sekce 2.4), jejichž výpočet osvětlení je někdy plně založen na zobrazování scény a nic jiné ho nebrzdí.

3.2 Práce pouze s viditelnými polygony

Při kreslení po polygonech bývá většinou použit z-buffer a množina polygonů se postupně vykresluje do color bufferu a z-bufferu. Nejjednodušší je vykreslit takto všechny plošky ve scéně. Ovšem těch může být o několik řádů větší množství, než skutečně viditelných plošek. Jako příklad poslouží budova, v každém jejím místě je viditelný pouze zlomek všech jejích polygonů. Významným přínosem tedy je umět rozhodnout které polygony jsou a které nejsou viditelné a kreslit pouze ty viditelné. Pro dynamické scény je to obtížnější než pro statické, ovšem i zde se objevily efektivní algoritmy, například [Sud96].

Tato optimalizace je přesně stejně jako minulá opět přínosná prakticky všude a největší užitek přinese systému založenému na rastrových metodách (popsány v sekci 2.4). Obě optimalizace lze zkombinovat.

3.3 Obrazová cache

Obrazy objektů vykreslených v jednom snímku můžeme uložit do obrazové cache (image cache) a použít beze změny i v dalším snímku, pokud analyticky spočítáme nebo odhadneme, že se nebudou lišit více, než o maximální povolenou chybu. Postup byl představen v [Sch96].

I pro obrazovou cache platí stejné závěry jako pro předchozí optimalizace. Rozdíl je pouze v tom, že víceprůchodové metody, pro které je vyrenderovaný obraz pouze meziprodukt určený pro další zpracování, nejsou tak necitlivé k chybám na úrovni pixelů jako lidský pozorovatel. Proto je při implementaci nutné provázat obrazovou cache se zbytkem systému tak, aby po nepatrném pohybu objektu nebo kamery zůstal identický nejen obraz objektu, ale pro účely výpočtu i jeho souřadnice vzhledem ke kameře.

Obrázek 3.1: Ilustrační scéna - „dveře“.

Podobný přístup použitý v oblasti raytracingu, tzv. render cache, byl popsán v [Wal99]. Jde též o způsob, jak znovuvyužít již dříve spočítaná data tak, aby nevznikla příliš velká chyba, render cache je ovšem svým návrhem svázána s oblastí raytracingu a do jiných oblastí přenést nelze.

3.4 Stupně detailu

Pro interaktivitu v komplexních scénách obsahujících velké množství detailů je použití různých stupňů detailu (level of detail, LOD) prakticky nepostradatelnou technikou. Myšlenka je jednoduchá - odstraněním detailů v geometrii

scény vznikne při výpočtu nebo zobrazení chyba, operace ale proběhne rychleji. Při odstranění vhodných detailů pak může být zrychlení dostatečně velké a chyba dostatečně malá na to, aby výkon celého systému vzrostl.

V praxi bývá před zahájením práce pro všechny objekty nebo nějaké části scény obvykle zkonstruována hierarchie různých stupňů detailu, během práce je pak podle okolností vybírán nejvhodnější stupeň.

V naší situaci, kdy nad modelem scény počítáme osvětlení a poté scénu zobrazujeme, mohou ještě nastat různé možnosti. V ideálním případě, pokud to použité metody výpočtu a zobrazení umožní, lze pro výpočet i zobrazení volit různé stupně detailu a výsledky výpočtu do zobrazovaného modelu transformovat. Jindy může být nutné použít pro oboje stejný model.

V každém případě nám ale stupně detailu umožní rozhodovat se mnohem svobodněji mezi rychlostí a kvalitou, při zvyšování rychlosti pak kvalita často zůstává dlouho vysoká. Složitost původní scény (nejjemnějšího stupně detailu) je přitom prakticky neomezená, systém může dynamicky nahrávat do paměti jen takové stupně detailu, které právě potřebuje a brát přitom v potaz i množství volné paměti tak, aby byla stále zachována interaktivita.

Zjednodušováním geometrie se v minulosti zabývalo mnoho prací a vzniklo mnoho desítek různých algoritmů [Hec97a]. Otázka které z nich a kdy jsou nejvhodnější ovšem zůstává otevřená a já se jí zde zabývat nebudu.

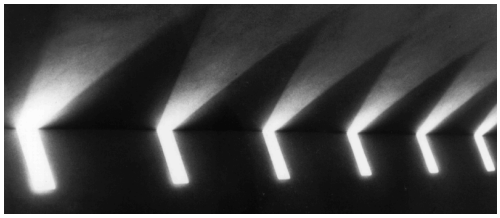
Propojení stupňů detailu s různými metodami výpočtu globálního osvětlení není vždy přímočaré a může přinést řadu obtížných problémů. Jejich vyřešení je ale nezbytným krokem před nasazením v opravdu komplexních dynamických scénách.

Do oblasti výpočtu globálního osvětlení bývají stupně detailu téměř vždy vneseny v podobě různých typů clusterů [Sil94, Smi94] (přehled v [Has99]). Vlastní typ clusteru a jeho implementaci popisuji v sekci 4.4.

Použití stupňů detailu se v obecném případě nevylučuje s předchozími optimalizacemi.

4

Návrh založený na střílení paprsků



The beams and bridges cut the light on the ground
into little triangles and the rails run round

Suzanne Vega, Ironbound

První v této práci navrhovaný přístup k výpočtu globálního osvětlení je založen na střílení paprsků ze zdrojů světla do celé statické scény (Monte Carlo radiosita, přehled v [Bek99]) a zvlášť směrem k dynamickým objektům.

Zatímco Monte Carlo radiosita ve statických scénách je poměrně jednoduchý postup využívající vystřelení jednoho paprsku k vyzáření jednotky energie (paprsek se odráží a předává svou energii zasaženým plochám), pro podporu dynamických scén jsou nutná určitá rozšíření. Ta popisuji v sekci 4.2.

Použitému modelu adaptivního dělení povrchů a interpolaci osvětlení je věnována sekce 4.3.

Sekce 4.4 pak popisuje clustery, které jsem pro tuto metodu navrhl a použil s cílem zvýšit kvalitu osvětlení v komplexních scénách i při vržení malého množství paprsků.

V sekci 4.5 je stručně vyložen stav implementace v době odevzdání práce a připojeno několik ukázek počítaného osvětlení.

Součástí návrhu a implementace je, kromě toho co bude popsáno, řada dalších optimalizací zrychlujících výpočet nebo snižujících paměťové nároky konstantně krát, ne asymptoticky. Jejich popis by musel zabíhat příliš hluboko do implementačních detailů a značně by se promítl do délky práce, proto jsem od něj upustil.

4.1 Vnější požadavky

Tato práce byla od počátku zaměřena na použití v reálných aplikacích a snažila se splňovat jejich požadavky.

Jedna z potenciálních aplikací jsou počítačové hry. Systém pro ně vyvinutý by měl dovolit pohyb a změny tvaru objektů, které mohou být i primárními zdroji světla a vznik a zánik objektů. Naopak realistické chování průhledných objektů není prioritou. Změny stupně detailů (level of detail, LOD) objektů při zobrazování nemusí výpočet osvětlení nijak ovlivnit, počítat lze nad jiným stupněm a výsledek při zobrazování vhodně transformovat.

Ve skutečnosti jsou ale enginy her prakticky výhradně založeny na tom, že nemusí řešit obecný případ, jejich vývoj se zaměřuje na co nejlepší model využití předpočítaných dat a co nejlépe vypadající a nejsnáze spočitatelné dynamické efekty. Takovým enginům pak systém počítající skutečné globální osvětlení, a to i ve zcela obecné scéně, nemůže konkurovat a ocenil by ho pouze malý okruh platících hráčů. Vývoj v obecných scénách se přesto později může promítnout i do této oblasti, proto má smysl brát požadavky her v potaz.

Ve většině dalších potenciálních aplikací může uživatel manipulovat se scénou svobodněji a navozovat různé nepříjemné situace, kdy například malá změna geometrie způsobuje významnou změnu osvětlení nebo kdy je většina scény ozářena odraženým světlem.

Zadáním je tedy řešit prakticky zcela obecný případ komplexních dynamických scén. Jediné, s čím lze počítat a co lze případně využít, je fakt, že většina scény bývá skoro vždy statická a v jeden okamžik se obvykle mění jen malá část geometrie.

4.2 Řešení vyčleněním dynamických objektů

Pokud očekáváme, že se v komplexní scéně bude naráz měnit jen malá část geometrie, efektivní implementaci velmi pomůže rozdělění celé scény na statické a dynamické objekty. Vhodné je dovolit přitom objektům přecházet z jedné kategorie do druhé. Osvětlení se pak počítá klasicky pro statickou scénu a odděleně od ní pro vliv dynamických objektů na celou scénu; tento přístup jsem také zvolil.

V následujícím textu budu základní elementy tvořící scénu nazývat plošky.

V mé implementaci jde o trojúhelníky, možné by ovšem bylo použít i obecnější prvky.

4.2.1 Statické objekty

Statické objekty nemění svůj tvar ani polohu ve scéně, jediná dovolená aktivita je změna výkonu v nich obsažených primárních zářičů. Taková změna se promítne do množství ze zářiče nevystřelené energie; ta je pak při nejbližší příležitosti vystřelena.

4.2.2 Dynamické objekty

Chování dynamických objektů není nijak omezeno, mohou se pohybovat, měnit tvar, množství plošek, množství a výkon primárních zářičů, mohou vznikat a zanikat. Bylo by dokonce možné všechny statické i všechny dynamické objekty považovat za dva objekty složitých tvarů. Smyslem dělení na více objektů je pouze usnadnění přidávání a odebírání objektů a zefektivnění reprezentace oblastí, kde se plošky nacházejí, jednotlivé objekty mohou mít svá obalová tělesa. Dobré přiblížení obalovými tělesy zvyšuje efektivitu výpočtů konfiguračních údajů, umožní nám vrhat paprsky pouze do směrů, ve kterých došlo ke změně osvětlení. Ve své implementaci jsem použil obalové koule.

4.2.3 Vyplývající požadavky

Z koncepce statických a dynamických objektů plyne nutnost efektivně implementovat

- výpočet a průběžné zpřesňování statického globálního osvětlení
- výpočet vlivu dynamických objektů na statické globální osvětlení s případným využitím výsledků z minulých snímků (vliv může být kladný v podobě osvětlení vyzářeným nebo odráženým světlem nebo záporný v podobě zastínění)
- přidání statického objektu
- odebrání statického objektu

- libovolné změny dynamických objektů

Jednotlivým požadavkům se budu věnovat v následujících sekcích.

4.2.4 Výpočet a zpřesňování statického osvětlení

Vliv statických objektů na globální osvětlení stačí spočítat jednou a případně později zpřesňovat. Každá ploška statického objektu zná své konfigurační faktory vůči ostatním ploškám statických objektů v hypotetické scéně bez dynamických objektů. K výpočtu konfiguračních faktorů z jedné plošky je použito všesměrové vystřelování paprsků. Směry jsou přitom voleny tak, aby každý paprsek nesl stejné množství energie. Vystřelené paprsky se mohou odrážet i dělit, ale nakonec vždy skončí předáním veškeré energie, zásahy se zatím ukládají do zasažených plošek. Po vystřílení veškeré energie určené k rozptylu následuje zpracování zasažených plošek (při nerovnoměrném pokrytí zásahy jsou děleny na jemnější elementy, řídké zásahy jsou naopak propagovány vzhůru do clusterů). Konfigurační faktory jsou pak odvozeny z množství energie předané v zásazích.

4.2.5 Vliv dynamických objektů na statické osvětlení

Vliv dynamických objektů by se měl počítat nebo aktualizovat v každém snímku, protože jakýkoliv i malý pohyb nebo změny tvaru mohou mít velký vliv na globální osvětlení. Zároveň jsme dynamickým objektům dovolili měnit počet plošek, takže obecně ani nelze udržovat mezivýsledky a dopočítávat pouze změny, například posuny stínů. Zdá se, že jsme dynamickým objektům dali příliš velkou volnost a způsobili si tak komplikace při výpočtu jejich účinků na globální osvětlení. Ve skutečnosti by nám ale omezenější dynamické objekty nepomohly, jinými autory popsané pokročilejší využívání mezivýsledků (např. dynamic discontinuity meshing) není na Monte Carlo radiositu přenositelné. Spokojíme se tedy s několika jednoduššími postupy.

Kostra algoritmu

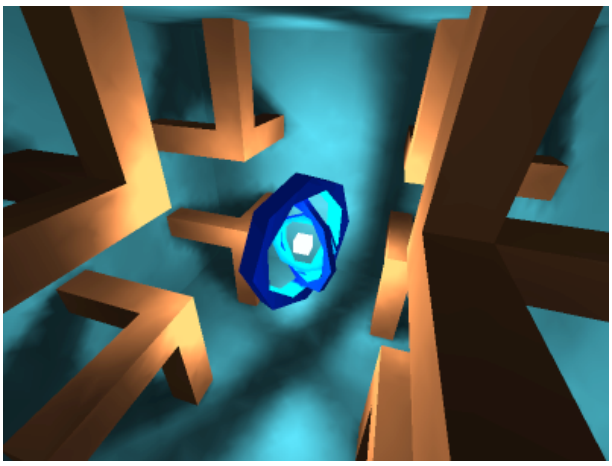
Na začátku máme statickou scénu s vypočteným globálním osvětlením a množinu dynamických objektů, jejichž vliv na globální osvětlení chceme spočítat.

1. Podle požadované přesnosti rozhodneme kolik energie ponese v následujícím cyklu každý jeden vystřelený paprsek, čím více energie, tím méně paprsků bude třeba vystřelit

2. Vybíráme významné dynamické zářiče a dvojice statický zářič, dynamický objekt. Z každého dynamického zářiče vystřílíme všechnu jeho energii do všech směrů, ze statického zářiče střílíme pouze směrem k dynamickému objektu, resp. jeho obalovému tělesu. Paprsky interagují se scénou, odrážejí se a ukládají do plošek pozice zásahů spolu s množstvím předané energie.

3. Projdeme všechny zasažené plošky a v případě nerovnoměrného rozložení zásahů je spolu se zásahy rozdělíme na jemnější elementy.

V tomto algoritmu se pro dynamické plošky nepočítají žádné konfigurační faktory, paprsky se odrážejí a energie distribuuje až k cílovým ploškám. Pouze při zásahu statické difusní plochy lze energii přidat do akumulátoru a po skončení střílení ji rozdistribuovat pomocí známých faktorů zasažené plochy.



Obrázek 4.1: Ilustrační scéna - „kola“.

Jak je vidět, na začátku snímku je nutné pevně zvolit kvalitu výpočtu. Flexibilnější by ovšem bylo kontinuální a kdykoliv ukončitelné zlepšování. Proto krok 2 opakujeme až do požadavku ukončení výpočtu; začínáme přitom s nízkou kvalitou (malým množstvím střílených paprsků) a s každým dalším průchodem ji zvyšujeme (řešení zpřesňujeme střílením dalších stále větších množství paprsků).

Výběr dynamického zářiče

Zářiče vybíráme podle celkového množství energie z nich šířeného, čím více energie, tím větší pravděpodobně bude vliv na globální osvětlení. Je možné přidat optimalizace pro velké scény, kde je řada zdrojů světla skryta mimo záběr a buď není třeba s nimi vůbec počítat nebo stačí nižší kvalita.

Výběr dvojic statický zářič, dynamický objekt

Pro každou dvojici *(významný statický zářič, dynamický objekt)* trvale udržujeme a aktualizujeme údaje o její důležitosti. Pokud se všechny paprsky střílené směrem k dynamickému objektu zarážejí o překážku, význam dvojice klesá a v dalších snímcích stačí nepatrné množství paprsků pouze pro kontrolu zda se objekt nedostal ze stínu. Význam dvojice mimo to roste se silou zářiče, klesá se vzdáleností objektu apod.

Směr paprsků

Z dynamických zářičů střílíme paprsky všesměrově, dokud nevystřílíme všechnu jeho energii. Ze statických zářičů střílíme paprsky k dynamickým objektům a ke všem lesklým plochám, ze kterých se k dynamickým objektům mohou odrazit.

V současné implementaci střílím pouze směrem obalových těles objektů, cesty s odrazem o lesklou plochu jsem vypustil. Střílení na obalové těleso je neefektivní v případě řídkých objektů; takové by si vyžádaly alternativní řešení, například se střílením po všech konvexních komponentách tvořících objekt.

Uložení energie v akumulátorech

Při šíření energie mezi ploškami je třeba v každé uchovávat jednak kolik energie ploška vyzařuje (kolik celkem přijala nebo už obsahovala a rozhodla se všesměrově vyzářit do okolí) a za druhé kolik energie má ještě vyzářit aby dosáhla rovnováhy mezi vytvářením+příjmem a absorpcí+výdejem energie. Oba akumulátory se upravují paralelně, energie z druhého ovšem může být, a je naším cílem aby byla, rozdistribuována do okolí. Při tom je druhý akumulátor vynulován, první zůstává nedotčen a nadále vyjadřuje celkové množství energie, kterou ploška vyzařuje (podle něj je při zobrazování plošky volen

její jas). Oba akumulátory jsou zdvojeny tak, aby šlo odděleně řešit statické globální osvětlení a vliv dynamických objektů na něj. Při zobrazení scény jsou statická i dynamická část sečteny a ovlivní jas plošky.

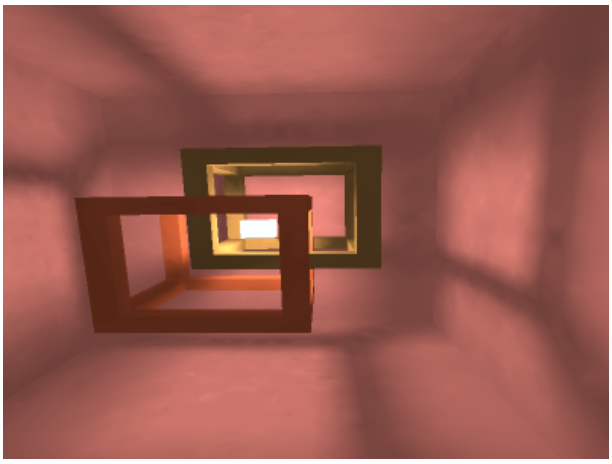
Chování paprsků

Aby byla veškerá přenášená energie správně započítána, je potřeba přiřadit různým paprskům různé chování.

Základní typ je statický paprsek používaný při výpočtu statického osvětlení. Dynamickými objekty volně prochází, od lesklých statických se může odrážet, při průchodu transparentním povrchem se může větvit, difusním plochám předává svou energii.

Paprsek vystřelený z dynamického emitoru se chová podobně, ovšem naráží i do dynamických objektů a odráží se i od difusních ploch (od statických difusních ploch nemusí, pro ně jsou známy konfigurační faktory a lze je využít).

Paprsek vystřelený ze statického zářiče směrem k dynamickému objektu mu při zásahu předá energii, kterou ho osvětlí. Z místa zásahu pokračuje odražený paprsek stejně jako z dynamického emitoru, mimo něj ale skrz těleso dál letí stínový paprsek se zápornou energií. Ten už interaguje pouze se statickou scénou, dynamickými objekty prochází. Jeho smyslem je vytvořit stín dynamického objektu a další jevy zastíněním způsobené.



Obrázek 4.2: Ilustrační scéna - „magic view“.

Využití výsledků z minulých snímků

Výsledkem výpočtu pro jeden snímek jsou hodnoty radiosity v jednotlivých ploškách. Hodnoty jsou zatíženy šumem, který můžeme snadno zmenšit (zmenšením DELENI, viz 4.3.1), ovšem za cenu zpomalení výpočtu. Zde se ovšem nabízí možnost snížit šum bez zpomalení, využitím hodnot z minulých snímků. Ve své implementaci jsem použil jednoduchý systém průměrování spočtené radiosity s hodnotou z minulého snímku. Při vahách 0.5 a 0.5 šumu výrazně ubyde a vznikající setrvačnost stínů ještě není příliš velká. Na experimentování se sledováním a využitím rozptylu radiosit už nedošlo.

4.2.6 Přidání statického objektu

Po přidání nového statického objektu je třeba

- osvětlit objekt - spočítat konfigurační faktory od zářičů z okolní statické scény k objektu
- vytvořit stíny objektu - upravit konfigurační faktory od zářičů z okolní statické scény k ploškám objektem zastíněným
- osvítit okolí vlastními zářiči objektu a odrazy od něj - spočítat konfigurační faktory od plošek objektu

První dva úkoly se řeší střílením paprsků z významných statických zářičů směrem k objektu a lesklým povrchům, stejným postupem, jaký byl popsán pro výpočet dynamického osvětlení. Výpočet potřebných konfiguračních faktorů od přidaného objektu ven do scény zařídí engine distribuující energii; faktory přepočítává až když je potřeba z plošky distribuovat významné množství energie a ta své faktory nezná nebo jen s přesností nedostačující pro současné množství přenášené energie.

4.2.7 Odebrání statického objektu

Po odebrání statického objektu nebo jeho přesunu mezi dynamické objekty je třeba

- zrušit ve statické scéně stíny vržené objektem
- zbavit statickou scénu všech konfiguračních faktorů vedoucích k ploškám objektu

Zrušení stínů vržených objektem

Každý objekt byl jednou mezi statické objekty zařazen a byl spočítán jeho vliv na scénu, stíny jím vržené. Kdyby byly statické objekty odebírány ve stejném pořadí, v jakém byly vloženy, bylo by triviální efektivně si zapamatovat stav scény před vložením objektu a při odebírání ho obnovit. To ovšem nelze po tvůrcích zobrazovaného světa požadovat, systém musí zůstat dostatečně flexibilní a umožňovat libovolné přidávání a odebírání. Princip skládání globálního osvětlení z množiny vlivů různých objektů by přesto šlo rozvinout a stíny pak odebírat velmi efektivně, implementace by ale byla velmi složitá a s počtem objektů by pravděpodobně kvadraticky rostla její zpočátku velmi malá paměťová a časová složitost. Jednodušší přístup počítá s odebíráním stínů stejným způsobem, jakým byly přidány, tedy zopakováním jejich výpočtu a oproti přidávání opačnou úpravou konfiguračních faktorů. Odebrání stínu je tedy stejně náročné, jako jeho přidání, efektivnější implementace s hypotetickou konstantní časovou složitostí by nemohla přinést více než zdvojnásobení rychlosti přidání a odebrání objektu, ve skutečnosti by to bylo ještě méně. Ovšem je třeba brát v úvahu ještě jeden vliv. Střílení je založeno na náhodě, takže po přidání a odebrání stínu nedostaneme stejné osvětlení, jaké zde bylo na začátku, dochází ke kumulaci chyb způsobených náhodným střílením.

Zrušení faktorů vedoucích k objektu

Je třeba zbavit statickou scénu všech konfiguračních faktorů vedoucích k ploškám rušeného objektu nebo je alespoň označit za neplatné a při prvním přístupu k nim je odstranit. Vzhledem k tomu, že přesuny objektů z množiny statických do množiny dynamických nejsou příliš časté, zatímco používání konfiguračních faktorů je časté velmi, je efektivnější odstranit faktory hned, než při každém použití jakéhokoliv faktoru testovat jeho platnost. Průchod množinou všech faktorů a odebrání neplatných urychlí, když si bude každý statický objekt, který smí být odstraněn, přímo pamatovat všechny faktory vedoucí k němu zvenčí nebo alespoň všechny plošky, které ho zvenčí osvětlují.

4.2.8 Změny dynamických objektů

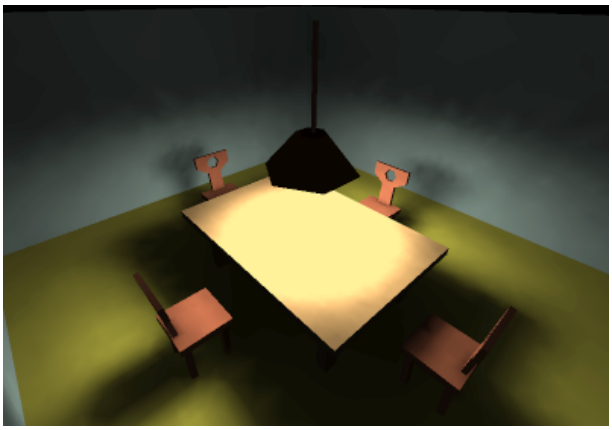
Dynamické objekty mohou libovolně měnit polohu, tvar, počet plošek, zářičů, v každém dalším snímku bude jejich vliv na osvětlení spočten znovu.

Takto obecné chování je možné díky použitému způsobu využití osvětlení z minulých snímků.

Při průměrování spočteného dynamického osvětlení s osvětlením z minulého snímku vznikne při velké změně dynamické geometrie velká chyba, během pár snímků ovšem zmizí. Při očekávané vysoké snímkové frekvenci je to přijatelné.

4.3 Adaptivní dělení a interpolace osvětlení

Cílem adaptivního dělení je umožnit dobrou aproximaci osvětlení i jednoduchými (např. pouze konstantními) funkcemi aniž by bylo nutné scénu předem připravovat a velké nebo jinak nevhodné plochy dělit na menší. Při použití pouze konstantních funkcí je nutné myslet zároveň na interpolaci osvětlení při zobrazování a přizpůsobit jí dělení.



Obrázek 4.3: Ilustrační scéna - „jídelna“.

Pro svou implementaci jsem zvolil reprezentaci scény trojúhelníky, jejich adaptivní dělení vždy na dva poloviční trojúhelníky a aproximaci osvětlení konstantami s interpolací při zobrazení.

Při dělení trojúhelníku je k rozpůlení vybírána nejdelší hrana. Postupným dělením se tak snižuje nepříznivý vliv polygonů s příliš ostrými úhly, jsou-li ve scéně nějaké.

Častým kritériem při dělení polygonů je rozdíl v osvětlení jejich vrcholů nebo gradient v osvětlení jejich povrchu [Han91], jako vylepšení pak bylo na-

vrženo vícestupňové dělení (multi-stage subdivision) [Pop00]. Pro svou implementaci jsem navrhl odlišný přístup.

4.3.1 Kritérium dělení

V průběhu výpočtu nebo zpřesňování konfiguračních faktorů z jednoho trojúhelníku vylétají paprsky a v ostatních trojúhelnících se hromadí souřadnice a váhy zásahů. Po vystřílení vhodného množství paprsků nastává propagace zásahů buď nahoru do clusterů (sekce 4.4) nebo dolů do jemnějších elementů a pokud takové neexistují, jsou dynamicky vytvořeny.

Kritériem pro propagaci dolů je poměr vzdálenosti průměrného zásahu od těžiště trojúhelníku násobené vahou zásahů k obvodu trojúhelníku. V mírně obecnější podobě v programu pak kritérium vypadá takto:

```
vzdalenost(teziste,prumernyZasah)*zasahu*DELENI>obvod
    && obvod>DOLNIMEZ
```

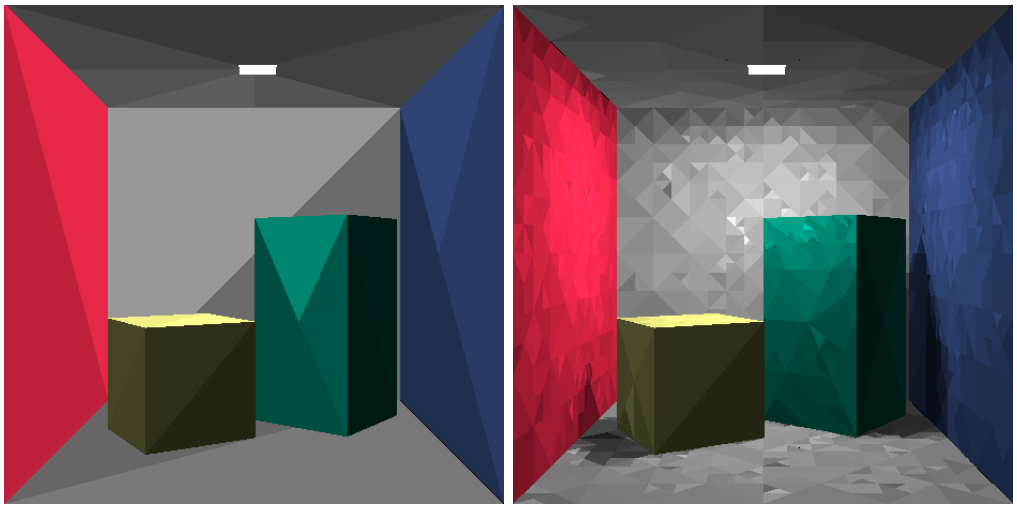
Pokud je poměr dostatečně vysoký (řízeno konstantou DELENI), zásahy jsou v trojúhelníku rozděleny příliš nerovnoměrně pro dobrou aproximaci konstantou a na místě je dělení. Jako nejvhodnější hodnota pro DELENI se ukázalo číslo *1*, při vyšších hodnotách je ve výsledcích vznikající šum už příliš výrazný i pro dynamické scény, nižší hodnoty zas zpomalují konvergenci.

Pro obvod trojúhelníku jsem zavedl i dolní mez DOLNIMEZ, za kterou už k dělení nikdy nedochází. Jde ovšem pouze o obranu proti degeneracím, které mohou vznikat vinou diskretizace reálných souřadnic zásahů do několikanásobně úspornější formy (jde o volitelnou optimalizaci, která několikanásobně snižuje paměťové nároky).

Navrhl jsem a experimentoval i s jinými složitějšími a teoreticky přesnějšími kritérii. Ovšem během dlouhého testování v mnoha různých scénách se toto jednoduché kritérium plně osvědčilo, přesnost osvětlení konvergovala vždy v celé scéně rovnoměrně a ke správnému řešení. Použití vícestupňového dělení [Pop00] by mohlo i u tohoto kritéria přinést zrychlení konvergence, to jsem ovšem zatím neověřil.

Vedle dělení příjemců (při nerovnoměrně rozmístěných zásazích) je nutné dělit ještě zářiče; není vhodné, aby z poloviny osvětlená plocha vyzařovala rovnoměrně celým povrchem, v takové situaci je potřeba spočítat konfigurační faktory z každé poloviny zvlášť a distribuovat energii odděleně. Pro

rozhodování zda zářič dělit jsem zvolil jednoduché kritérium, porovnávám množství energie vyzařované oběma syny zářiče a při významném nepoměru zářič dělím. Na obrázku 4.4 je v jednom okamžiku zachyceno dělení scény na zářiče a dělení na příjemce. Linky ovšem nevedou pouze mezi naznačenými trojúhelníky, ale i z a do větších celků, zobrazeno je nejjemnější dosud použité dělení (zachyceno zhruba po dvou sekundách od zahájení výpočtu na K6-2 500 MHz).



Obrázek 4.4: Hrubší dělení zářičů, jemnější dělení příjemců.

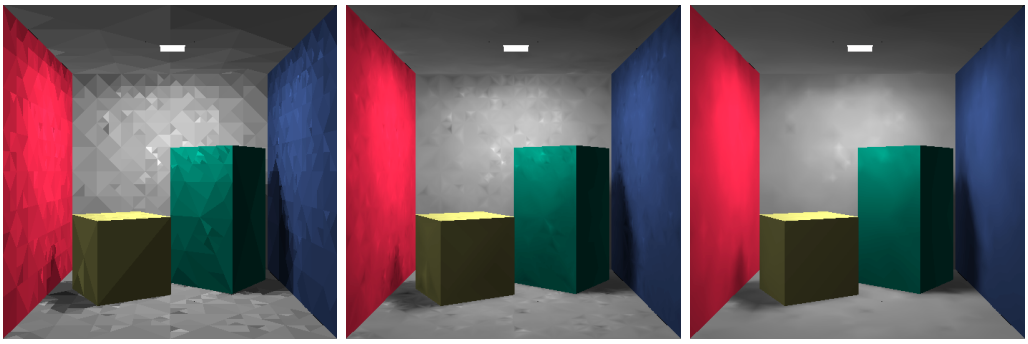
4.3.2 Eliminace T-vrcholů

Při každém dělení trojúhelníku vzniká nový vrchol, který se potenciálně může stát tzv. T-vrcholem způsobujícím chyby při interpolaci. Pro každý nový vrchol jsou tedy ihned rekurzivně provedena všechna nezbytná další dělení tak, aby T-vrchol nevznikl.

4.3.3 Interpolace

V popisované implementaci lze přepínat mezi zobrazováním bez interpolace a dvěma režimy interpolace (obr. 4.5). Jelikož radiosita je na začátku známa pouze pro trojúhelníky a ne pro vrcholy, radiositu vrcholů počítám jako vážený průměr radiosit všech elementů (clusterů, trojúhelníků i podtrojúhelníků) vrchol obsahujících. Váha elementu je přitom úměrná tomu jak velká

část okolí vrcholu je součástí elementu. Taktéž rozlišuji hrany a vrcholy, které je třeba interpolací skrýt, a ty, které mají zůstat viditelné, pak do seznamu elementů podílejících se na radiositě vrcholu zařazuji jen vhodnou podmnožinu elementů a jeden vrchol může mít několik různých seznamů. Seznamy elementů a jejich váhy jsou spočítány už při konstrukci clusterů a pak aktualizovány při dělení trojúhelníků, spočítané vážené průměry jsou pak pro maximální efektivitu cachovány a aktualizovány pouze pokud je nutné je použít a údaj v cache byl při změně radiosity nějakého elementu označen za neplatný.



Obrázek 4.5: Režimy interpolace

Samotná interpolace osvětlení pak může probíhat mezi existujícími vrcholy, jejichž radiosity samy už vznikly průměrováním, redukce šumu je tedy vyšší. Druhá varianta je přidat každému trojúhelníku pouze pro účely zobrazování prostřední vrchol, za jeho radiositu považovat spočítanou radiositu trojúhelníku a interpolovat mezi všemi čtyřmi vrcholy. Redukce šumu je přitom slabší (viz. prostřední obrázek v 4.5).

Oba způsoby interpolace přitom slouží pouze optickému zlepšení výsledků, do výpočtu se nijak nepromítají. V rámci redukce šumu mohou dočasně (než výsledek lépe zkonverguje) odstranit i některé správně spočítané detaily, v praxi jsem ovšem tento jev pozoroval jen v nevýrazné podobě. Na druhou stranu lze většinou předpokládat, že interpolací v rychle a nepřesně spočítané scéně se o velký kus přiblížíme tomu, jak skutečné osvětlení má vypadat a po delší konvergenci by vypadalo. Interpolace je tedy velice důležitou součástí celého systému.

4.4 Clustery

Clustery [Sil94, Smi94] jsou známé a při implementaci radiačních metod používané objekty vycházející z principu různých stupňů detailu (LOD). Jeden cluster je určitou aproximací množiny polygonů, podle jeho typu různě přesnou a výstižnou. Obvykle mohou clustery dobře aproximovat pouze množinu blízkých podobně orientovaných vzájemně se nestínících polygonů, existují ale i typy zachycující průhlednost v řídkých clusterech nebo zastoupení různých směrů v clusteru. Přehled existujících typů a zejména metod konstrukce clusterů je v [Has99].

Pro svou implementaci jsem navrhl a použil nový typ. Cluster zde též reprezentuje množinu polygonů, konkrétně trojúhelníků, ale na rozdíl od klasických clusterů nehraje roli při zjednodušování geometrie scény, slouží výhradně lepšímu využití informací z vystřílených paprsků, efektivnějšímu propojování ploch linky v hierarchické radiositě (sekce 4.4.3) a rovnoměrnějšímu rozprostírání přijaté energie (sekce 4.4.4).

4.4.1 Konstrukce

Clustery konstruuji shora dolů tak, aby jejich hierarchie tvořily binární stromy:

1. Každý objekt s alespoň dvěma trojúhelníky budiž clusterem.
2. Každý cluster rozděl podle kritérií optimality na dvě části a z těch, které obsahují alespoň dva trojúhelníky, udělej další clustery.

Optimální dělení je podle mého návrhu takové, při kterém

- jsou od sebe odděleny skupiny odlišně orientovaných trojúhelníků
- jsou od sebe odděleny skupiny vzdálených trojúhelníků
- je cluster rozdělen na stejně velké části.

4.4.2 Organizace do binárních stromů

Organizace clusterů do binárních stromů má svůj význam při abstrahování od typu elementu, i jednotlivé trojúhelníky dělím při adaptivním zjemňování na poloviny, celý objekt pak tvoří jeden binární strom a algoritmy nad

ním operující většinou nepotřebují vědět, zda pracují s clusterem, jedním trojúhelníkem nebo jeho částí.

4.4.3 Vystřelování z clusterů

První jednodušší uplatnění clusterů je ve zrychlení výpočtu konfiguračních faktorů vedoucích z trojúhelníků tvořících cluster. Pokud chceme vystřelením N paprsků spočítat všechny konfigurační faktory z K plošek, klasický přístup by byl vystřelit z každé plošky část paprsků a faktory odvodit nezávisle na sobě. Pokud vystřílíme všechny paprsky z celého clusteru a určíme tak společné faktory z clusteru, výsledek bude v případě alespoň trochu dobrého clusteru znatelně přesnější, resp. bude stačit méně paprsků. I pozdější případné přesuny energie skrz jeden link z clusteru jsou rychlejší než přesuny skrz množinu linků ze všech polygonů v clusteru.

4.4.4 Zásahy do clusterů

Druhé uplatnění clusterů je v zachytávání zásahů a rozprostírání zachycené energie do vhodně velkého okolí zásahu. Vzhledem k tomu, že mé clustery nezjednodušují geometrii scény, při hledání průsečíku je nalezen přímo zasažený trojúhelník a zásah uložen v něm. Všechny zasažené trojúhelníky jsou přitom evidovány (při prvním zásahu přidány do dynamicky rostoucího pole). Poté, po vystřílení všech paprsků pro daný snímek a nahromadění všech zásahů v trojúhelnících, následuje fáze propagace ojedinělých zásahů binárním stromem vzhůru do clusterů. Motivace je tato: pokud byl trojúhelník (resp. cluster) zasažen jen jednou nebo párkrát a jeho bratr ve stromě ani jednou, pravděpodobně to není tím, že by byl bratr jinak osvětlen, spíš jde o náhodu běžně nastávající při vystřílení příliš malého množství paprsků. Proto jsou zásahy v takové situaci převedeny ve stromu na otce, tedy na cluster sdružující zasažený objekt a jeho bratra. To se následně projeví rovnoměrným rozprostřením osvětlení po celém clusteru místo osvětlení koncentrovaného v jednom trojúhelníku. Při rozhodování zda je počet zásahů dostatečně malý pro propagaci vzhůru je přihlédnuto k relativním velikostem obou bratrů.

Za úspěch považuji implementaci propagace zásahů vzhůru se složitostí $O(t + p)$ kde t je počet zasažených trojúhelníků a p počet jejich předků v binárním stromě, šlo o poměrně netriviální problém.

4.4.5 Shrnutí významu clusterů

Význam clusterů silně roste s velikostí scény. Přesto se příznivě promítly i do rychlosti výpočtů v malých scénách do tisíce trojúhelníků. Po přesunu zájmu směrem k perspektivnějším rastrovým metodám jsem nicméně celé řešení založené na střílení paprsků opustil a neprováděl už žádná měření účinku těchto clusterů na rychlost konvergence. Očekávám, že rastrové metody přinesou lepší výsledky.

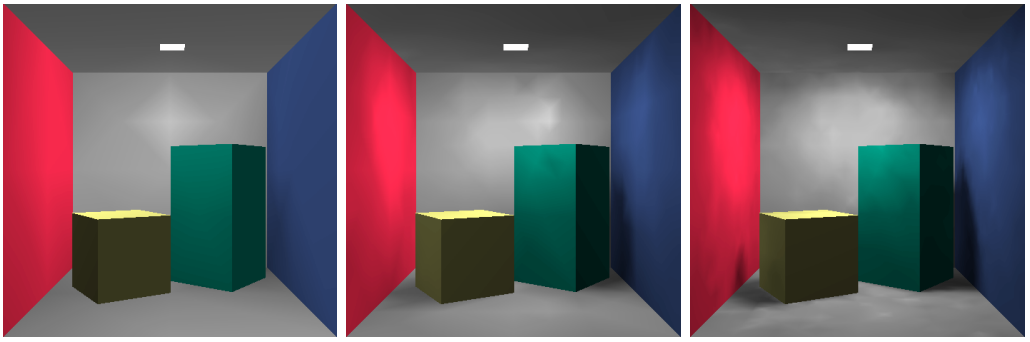
4.5 Stav implementace

4.5.1 Vnější rysy

Systém umožňuje práci s dynamickými scénami a to buď s ručním ovládáním objektů nebo s jejich pohybem po splajnových drahách, které mohou být součástí scény.

Scénou lze vždy procházet, probíhající výpočty nikdy nebrání interakci a to ani v DOSu, prostředí singletaskovém a singlethreadovém.

Podporované platformy jsou DOS (DJGPP), Linux (GCC), přenos na ostatní by neměl být obtížný.



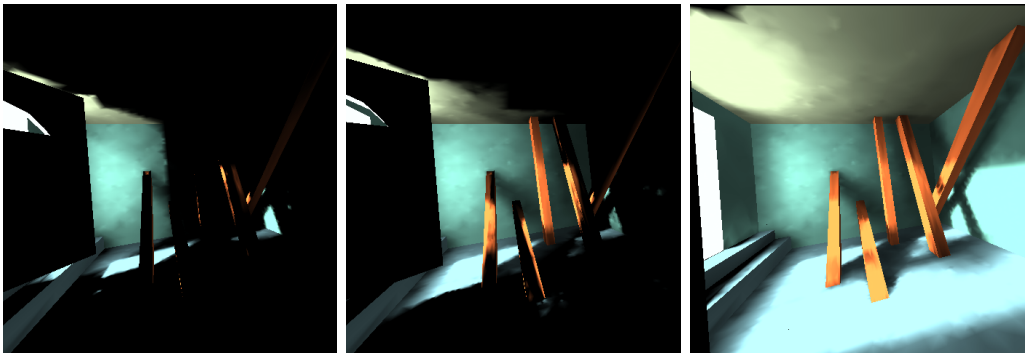
Obrázek 4.6: Statická scéna - stav po 0.1, 1 a 10 sekundách

(všechny uváděné doby výpočtu se týkají procesoru AMD K6-2 500MHz)

Ve statické scéně se osvětlení během procházení scénou neustále zpřesňuje (obr. 4.6). V dynamické scéně jsou možné tři režimy:

- zpřesňuje se pouze statické osvětlení, dynamické objekty do něj nejsou zahrnuty
- osvětlení se počítá pro každý snímek zcela odděleně, do výpočtu jsou zahrnuty statické i dynamické objekty (obr. 4.7)
- podle metody v této kapitole popsané se počítá pouze dynamické osvětlení, při zobrazování se skládá s dříve spočítaným statickým osvětlením (obr. 4.8)

Obrázek 4.7: Dynamická scéna při 2fps - každý snímek počítán zvlášť



Obrázek 4.8: Dynamická scéna při 2fps - aktualizován vliv dynam.objektů

Zobrazovací subsystém může využívat OpenGL nebo trojúhelníky rasterizovat sám. V druhém případě je ve scéně podporována jedna lesklá plocha (při 100% lesklé odrazivosti zrcadlo).

Mezi podporované vlastnosti materiálů patří difusní odrazivost, lesklá odrazivost, průhlednost, index lomu. Při průchodu paprsku scénou se všechny

vlastnosti uplatňují, paprsek se podle potřeby vhodně odráží nebo větví, výsledkem jsou tedy tzv. rozšířené konfigurační faktory. Zobrazovací subsystém ovšem v zájmu rychlosti zobrazuje jen difusně odražené světlo. Jinak řečeno, systém počítá libovolnou řadu lesklých a difusních odrazů a průchodů materiálem, závěrečný odraz směrem do kamery je ovšem vždy difusní.

Rozšíření k plné obecnosti s využitím střílení paprsků z kamery není složité a i s ním jsme experimentovali, značný pokles rychlosti ovšem neumožňoval interaktivní práci; ta by byla možná až po řadě dalších optimalizací raytracingu. Významný přínos při zobrazování raytracingem by byl využít ke zrychlení šíření paprsků od pozorovatele již dříve spočítané cesty šíření světla od zdrojů, jak ukazuje [Gra00]. Implementace podobného systému by ale stála v dané situaci příliš mnoho času.

Scénou se šířící světlo je vždy bílé, výslednou barvu dává objektům až závěrečný difusní odraz do kamery. Rozšíření ke korektnímu šíření barevného světla by bylo zcela nekomplikované, zvýšily by se pouze paměťové nároky a klesla rychlost. Jelikož barva světla ve většině reálných scén nehraje velkou roli, zůstal jsem prozatím u bílého světla.

4.5.2 Vnitřní rysy

Přidávání a odebírání dynamických objektů je možné kdykoliv. Po jakýchkoliv operacích se statickými objekty je nutné přepočítat statické osvětlení; aktualizace statického osvětlení, jak jsem je popsal, jsem už neimplementoval.

V současné implementaci jsou zlepšování statického a dynamického osvětlení na sobě nezávislé procesy volané funkcemi (metodami) improveStatic a improveDynamic. Při jejich volání není nutné určovat předem požadovanou kvalitu nebo dobu výpočtu, jejich parametrem je callback funkce, která zodpovídá otázku zda ve zlepšování pokračovat nebo skončit. Je tedy možné stanovit libovolné kritérium, zlepšovat dokud neuplyne určitá doba, dokud není dosažena určitá kvalita, dokud uživatel nestiskne klávesu apod. Callback je během výpočtu volán dostatečně často, takže na stisk klávesy je možná rychlá odezva.

Jádro zlepšování statického osvětlení vybírá vhodné emitory, distribuuje skrz ně naakumulovanou energii a většinou přitom zpřesňuje použité konfigurační faktory. Zlepšování jde kdykoliv přerušit a později v něm pokračovat nebo ho ukončit.

Jádro zlepšování dynamického osvětlení pracuje v rámci jednoho snímku v průchodech polem všech dvojic *(významný statický emitor, dynamický ob-*

jekt). Pro každou dvojici je známa a průběžně aktualizována její důležitost. Během průchodu jsou z emitorů na objekty střílena množství paprsků odpovídající důležitosti dvojice, a to tak, aby bylo během celého průchodu vystříleno K paprsků. K přitom s dalšími průchody geometricky roste. Zlepšování tak lze přerušit i po velmi krátké době a vždy bude k dispozici korektní výsledek s přesností odpovídající době výpočtu; náklady na opakované procházení polem přitom neohrozí výkon celého systému. Do dynamického osvětlení je zahrnuto i světlo odražené od dynamických objektů, na implementaci vystřelování z dynamických emitorů už ale nedošlo.

Výpočet dynamického osvětlení zůstal nedoladěn a obsahuje chyby, které se už nepodařilo odstranit. Pro obrázek 4.8 jsem vybral snímky, kde není vliv chyb výrazný a výsledek se blíží očekávání, jindy ale mohou být výsledky horší. Pokud je v dynamické scéně počítán každý snímek nezávisle (obr. 4.7), výsledek je korektní.

4.6 Shrnutí

Navrhl jsem a ve spolupráci s Danielem Sýkorou implementoval experimentální systém pro manipulaci s globálně osvětlenými dynamickými scénami.

Jádro počítající globální osvětlení vychází z hierarchické Monte Carlo radiosity, která jediná nabízí rychle konvergující aproximaci osvětlení prakticky okamžitě po zahájení výpočtu a dokáže ji kontinuálně zpřesňovat. Rychlost základní metody přesto při velkém nárůstu složitosti scén citelně klesá, proto jsem pro ni navrhl clustery, které vliv složitosti scény na rychlost konvergence snížily. Monte Carlo radiositu jsem poté adaptoval pro dynamické scény.

Ačkoliv Monte Carlo radiosita již byla známa, popisovaný systém byl vyvinut nezávisle a obsahuje oproti jiným řešením další originální a přínosné prvky, například kritérium pro dělení ploch.

Všechny algoritmy jsem navrhoval s ohledem na jejich časovou složitost tak, aby (až na hledání průsečíku paprsku se scénou) nezávisely na velikosti scény, neměly žádnou významnou počáteční režii, osvětlení zpřesňovaly kontinuálně a šly kdykoliv přerušit. Výsledný systém tak dokáže aproximovat osvětlení kdykoliv, i bezprostředně po zahájení výpočtu. Interakce je tedy možná i v komplexních scénách, s narůstající složitostí pouze klesá rychlost konvergence, resp. kvalita dynamicky počítaného osvětlení.

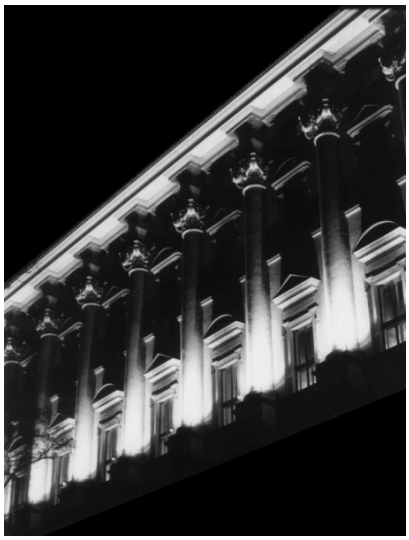
Systém byl prakticky nasazen v jednoduchých scénách, jak ukazují ilustrační obrázky a dema Pinokijo a Realtime Radiosity 2 dostupná např. na

http://movsd.scene.cz. Jedinou překážkou okamžitého nasazení v komplexních scénách je pomalý BSP generátor (jde o komponentu používanou při vytváření scén a nezávislou na zbytku systému). Lze ovšem kvalifikovaně odhadnout, že interaktivní práci s globálním osvětlením by ani zde nic nebránilo, pouze kvalita by byla nižší.

Největší nevýhodou systému je, stejně jako u všech Monte Carlo řešení, šum ve spočítaném osvětlení. Ačkoliv šum lze různými technikami snižovat nebo místně zcela odstraňovat, rozhodl jsem se začít se zabývat přednostně metodami, jejichž chyby na lidského pozorovatele působí méně rušivě. Po deseti měsících vývoje, v době kdy nebyl systém ještě zcela doladěn, vývoj dokončen a výsledky změřeny, jsem tedy Monte Carlo metody opustil a vydal se směrem hardwarově akcelerovaných stínových map.

5

Návrh založený na stínových mapách



O intenzitě polostínu rozhoduje velikost složky světelného vektoru, která projde analyzátorem. Tato složka, jak z výkladu Laurentovy desky vychází, záleží na úhlu, který svírá dopadající světelný vektor se směry obou kolmých složek, na něj se paprsek dělí v dvojlomné desce.

Teyssler-Kotyška, Technický slovník naučný

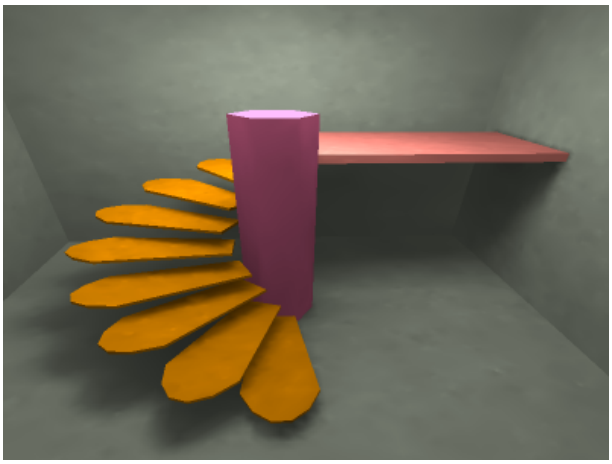
Druhý navrhovaný algoritmus vychází z rastrových metod, které jediné plně využívají možností 3D akcelerátorů a redukují nároky na procesor. Jedinou se současným běžným hardwarem plně akcelerovatelnou rastrovou metodou (zatěžující zejména akcelerátor namísto procesoru a sběrnice) jsou stínové mapy a ty jsem také použil.

Algoritmus jsem vzhledem k dříve uvedeným skutečnostem (sekce 1.2) stihl do doby odevzdání diplomové práce pouze popsat, nicméně na implementaci a dalším vývoji pracuji.

Algoritmus je založen na využití stínových map, nezáleží zda hloubkových či indexových. Ale protože indexové mapy jsou efektivně hardwarově implementovatelné na širším okruhu akcelerátorů, zabýval jsem se přednostně jimi. Implementaci indexových map s pomocí standardního OpenGL bez proprietárních rozšíření představím v sekci 5.1.

Zatímco artefakty vznikající při použití hloubkových map jsou dostatečně podrobně popsány, včetně způsobů jak je eliminovat, u indexových map jsem

na podobný zájem a práce nenarazil. Hned poté, v sekci 5.2, tedy artefakty proberu a navrhnu jejich ošetření.



Obrázek 5.1: Ilustrační scéna - „schody“.

Následovat bude návrh rozšíření stínových map k výpočtu globálního osvětlení a řada optimalizací směřujících k nasazení v komplexních dynamických scénách, ve scénách osvětlených výhradně odraženým světlem apod.

5.1 Indexové stínové mapy v OpenGL

Všechny práce týkající se implementace indexových stínových map v OpenGL, se kterými jsem se setkal, jsou založeny na využití proprietárních rozšíření OpenGL. Zde předvedu, že podobný výsledek je možný i s OpenGL bez rozšíření, jediný požadavek je podpora alfa kanálu.

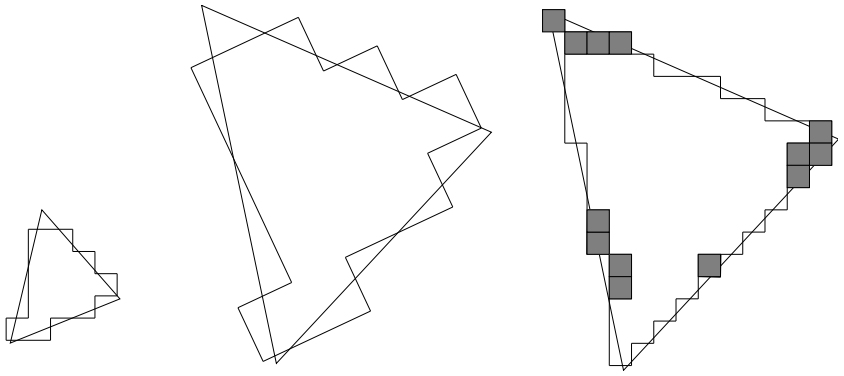
Indexy ve své implementaci ukládám do alfa kanálu, stínová mapa pak může obsahovat pouze alfa kanál, další složky jsou nadbytečné. K porovnání indexů slouží tzv. alfa test [Seg99]. Pokud je indexů více než hodnot, které lze uložit do alfa kanálu, je nutné generovat několik indexových map, z nichž každá obsahuje část indexu, a pro každý pixel pak provádět stejné množství alfa testů s jednotlivými částmi indexu. Je tedy vidět, že nižší počet bitů v alfa kanálu výpočet zpomaluje a na starších akcelerátorech bez alfa kanálu je nemožný. V dnes prodávaných akcelerátorech pro osobní počítače je běžných osm bitů, což pro efektivní implementaci postačuje.

(Pozn. Jak je vidět, je žádoucí mít indexů co nejméně. Všechny polygony ležící v jedné rovině mohou mít bez újmy na obecnosti a bez dalších nároků stejný index. Polygony tvořící konvexní objekt nebo konvexní komponentu mohou mít společný index jen pokud pro každý polygon ověříme, zda není od zářiče odvrácen a kreslíme ho v takové situaci neosvětlený.)

5.2 Artefakty indexových stínových map

5.2.1 Mizející krajní pixely

Nejvýznamnějším problémem při hardwarové implementaci indexových stínových map jsou mizející pixely podél hranic polygonů (obr. 5.2).



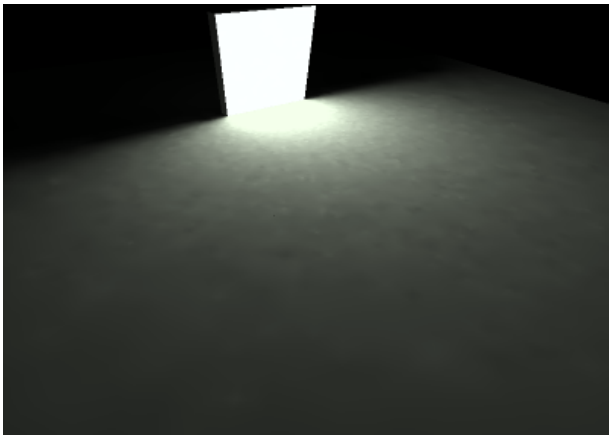
Obrázek 5.2: Mizející pixely
Vlevo možná rasterizace polygonu ve stínové mapě. Uprostřed: tentýž polygon je třeba vykreslit z jiného pohledu a zvětšený, naznačen tvar textury ve stínové mapě. Vpravo výsledek texturování s použitím stínové mapy, některé krajní pixely byly načteny ze sousedních polygonů ve stínové mapě a jsou proto považovány za zastíněné těmito polygony.

Ideální řešení by bylo, kdyby 3D akcelerátory dokázaly při texturování sledovat hranice polygonu v textuře a požadavky o načtení texelu vně hranic nahradily načtením nejbližšího vnitřního texelu. Oproti běžnému texturování by se tak změnilo pouze chování v oblasti problematických krajních pixelů a výkon by nemusel výrazně klesnout. Všechny potřebné informace má akcelerátor v době texturování k dispozici. O žádném akcelerátoru toto

podporujícím ovšem nevím, popíši tedy způsoby, jak lze problém za cenu několikanásobného zpomalení a případného drobného zkreslení řešit.

Pro jednoduchost se budu zabývat pouze scénami s uzavřenými objekty (takovými, že každou hranu sdílí právě dva polygony a vidět jsou pouze vnější strany polygonů), případné rozšíření všech popisovaných metod pro zcela obecné polygonové scény je ovšem snadné.

Nejprve je třeba rozdělit chyby do dvou kategorií. Chyby vznikají vždy na hraně polygonu, tato hrana přitom může být v rámci celého zobrazovaného objektu vnitřní nebo vnější. Vnitřní hrana je taková, kterou sdílí dva polygony ke kameře přivrácené nebo dva odvrácené. Vnější hranu sdílí jeden polygon přivrácený a jeden odvrácený. Vnější hrany lze též charakterizovat tak, že právě je provázejí diskontinuity v z-bufferu. Následující postupy mohou být vhodné jen pro některou z uvedených kategorií.

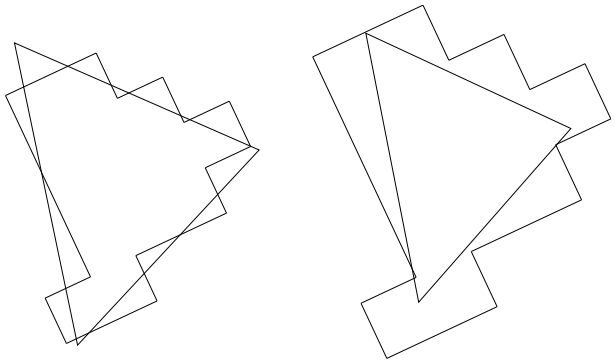


Obrázek 5.3: Ilustrační scéna - „testovací“.

První řešení určené pro obě kategorie je testovat při texturování jednoho pixelu ne jen jeden texel, ale i 8 nejbližších (popř. 4, 12, 24 apod., podle nutnosti opravit co nejvíce mizejících pixelů a podle přípustného zkreslení výsledku) a test shody indexu považovat za úspěšný uspěje-li alespoň jeden z 9 (resp. 5, 13, 25) testovaných texelů. To lze na akcelerátoru implementovat tak, že každý polygon vykreslíme 9–krát, pokaždé se souřadnicemi v textuře posunutými směrem k jednomu z nejbližších texelů. Využíváme přitom toho, že pixely, které neuspějí v (alfa) testu, akcelerátor nerenderuje; lze tedy texturovat víckrát přičemž ve výsledku se objeví právě pixely, které v testu alespoň jednou uspěly. Implementace je přitom velmi jednoduchá a nevyžaduje opakované přenosy geometrie scény po sběrnici, ta může být uložena

v akcelerátoru a procesor pouze 9–krát dodá změněnou matici pro generování souřadnic v textuře.

Druhé řešení pro obě kategorie chyb je modifikovat souřadnice polygonu v textuře tak, aby popisovaly polygon o půl texelu zúžený ve všech směrech (obr. 5.4). Pak nebude při texturování nikdy použit texel vně polygonu. Oproti předchozímu řešení stačí pouze jedno vykreslení, ovšem korektní zužování polygonu o půl texelu standardní generátor souřadnic z OpenGL nedokáže, musí ho provádět procesor a všechny souřadnice při vykreslování posílat akcelerátoru. Jedno takové vykreslení scény s generováním souřadnic na procesoru pak může být pomalejší, než 9 vykreslení v předchozí metodě.



Obrázek 5.4: Posun souřadnic v textuře (stínové mapě)

Je tu i možnost oba přístupy zkombinovat, pro každý polygon použít ten vhodnější. Pro polygony, které v textuře zabírají velkou plochu, může být rychlejší a přesnější zúžení o půl texelu, pro drobné polygony je vhodnější texturovat 9–krát.

Ani jeden z přístupů ovšem neeliminuje všechny chyby. Zužování není vhodné u polygonů s velmi ostrým úhlem u některého vrcholu, souřadnice vrcholu se zúžením posunou o příliš velkou vzdálenost a texturování je pak oproti tomu co očekáváme příliš zkreslené. Testování sousedních texelů zas rozšiřuje osvětlenou oblast o jeden texel všemi směry. Většinou nejde o viditelnou vadu, ve velmi řídkých případech to ovšem újmu způsobit může.

Řešení pouze pro vnitřní hrany představuje testovat každý texel ne pouze proti indexu kresleného polygonu, ale i proti indexům všech sousedních polygonů (implementace opakovaným vykreslením polygonu, pokaždé s jiným alfa testem). U uzavřených objektů se sousedící polygony nemohou zastiňovat, přístup je to tedy korektní.

U konvexních objektů lze všechny vnitřní hrany ošetřit tak, že pro celý

objekt použijeme jeden společný index. Při použití jednoho indexu pro nekonvexní objekt by tento nemohl vrhat stín sám na sebe. V každém případě zůstává problém vnějších hran, je tedy nutné kombinovat tento přístup s některou z předchozích metod.

5.2.2 Mizející polygony

Extrémním případem předchozího problému je zmizení celého polygonu v situaci, kdy mu při rasterizaci do textury nepřipadl žádný pixel. To se stává pouze drobným polygonům, ve výsledku ovšem drobné a touto chybou neosvětlené polygony působí značně rušivě.

Jednoduché řešení je zvýšit rozlišení textur (stínových map), tím lze mnoho podobných případů eliminovat.

V situaci, kdy by nebylo přípustné zbývající chyby u nejdrobnějších polygonů tolerovat, je možné následující řešení: Po vykreslení světelné mapy ověřit které polygony rasterizací zcela zanikly a otestovat viditelnost jejich vrcholů porovnáním jejich souřadnic z s údaji z příslušných míst z-bufferu podobně jako to dělá metoda hloubkových stínových map pro každý pixel. Na základě zjištěné viditelnosti vrcholů pak lze polygon kreslit celý osvětlený nebo zastíněný, při částečném osvětlení interpolovat mezi vrcholy.

5.2.3 Aliasing na hranicích stínů

Jelikož tvary stínů jsou uloženy v indexové mapě s nějakým konečným rozlišením, při jejich promítnutí do scény, při kterém dojde ke zvětšení, může vzniknout viditelný aliasing. Pro indexové mapy pravděpodobně neexistuje způsob jak tento jev bez značného zpomalení potlačit. Naštěstí nejde o závažnou chybu, skládáním více ostrých stínů vznikne řada méně výrazných aliasovaných přechodů mezi stupni stínu, které už dohromady nepůsobí tak rušivě jako jedna výrazná aliasovaná hranice ostrého stínu.

Aliasing lze kromě toho jednoduše snížit zvýšením rozlišení stínových map.

5.2.4 Neosvětlená část prostoru

Při použití jednoho nebo dvou rastrů místo polokrychle není stínovou mapou pokryt celý poloprostor, do kterého světlo z bodového zdroje směřuje. Kromě samozřejmé možnosti použít raději časově náročnější polokrychli nebo tři na sebe kolmé rastry pokrývající celý poloprostor se můžeme spokojit s tím, že část prostoru zůstane neosvětlena. Jde o směry, kterými vychází nejmenší množství světla, takže chybu lze obvykle v zájmu rychlosti tolerovat. Podle očekávané velikosti chyby se ovšem můžeme pro každý bodový zdroj zvlášť rozhodnout který typ mapy použijeme (jeden rastr, polokrychle apod.)

5.3 Výpočet globálního osvětlení

Jak bude vypadat navrhovaný systém počítající globální osvětlení (pouze difusní odrazy) v dynamických scénách pomocí stínových map? Následující kroky popisují výpočet jednoho snímku.

1. Zvol N podle požadavků na rychlost/kvalitu. Vyšší N znamená vyšší kvalitu a nižší rychlost.

2. Aproximuj všechny plošné primární zářiče ve scéně N bodovými zdroji s normálou (s maximálním vyzařováním ve směru normály a cosinovým útlumem s odklonem od ní). Pro zářiče s menší plochou stačí méně vzorků, pro slabé zářiče taktéž. Ani příliš mnoho vzorků na jeden zářič není třeba, i když je velmi významný. Každému bodovému zdroji je přitom přiřazeno určité množství energie a to tak, aby jejich součet odpovídal součtu energií všech plošných zářičů. Žádná energie díky tomu nezůstane nevystřelena.

3. Vygeneruj pro každý bodový zdroj stínovou mapu. Pokud pracujeme s hloubkovými mapami, nechť kromě hloubkové složky obsahují i indexy. Každý polygon nechť má unikátní index.

4. Vyber ze stínových map dohromady K vzorků (vzorek je jeden pixel ve stínové mapě, má tedy svůj index a lze určit jeho 3D souřadnice) a to tak, aby každý co nejlépe reprezentoval přibližně $1/K$ celkové energie zdrojů. Z map odpovídajících zdrojům s větším množstvím energie tedy vyber více vzorků, uvnitř map vybírej tak, aby každý vzorek dobře reprezentoval své okolí a tím procházela žádaná část energie. Každému vzorku přiřaď jakou energii reprezentuje (dosáhnout u všech přesně

$1/K$ nelze). Vzorky by neměly být rozmístěny pravidelně. To vše ovšem prováděj jen tak přesně, aby to nezpomalilo celý výpočet, nepřesnosti v této fázi výsledek nijak dramaticky neovlivní.

5. Rozděl vzorky podle indexů, pro každý zastoupený index sečti energie přiřazené vzorkům s tímto indexem. Výsledkem je hrubý náhled na to kolik který polygon přijal energie od primárních zářičů. Se zvyšováním K můžeme náhled podle potřeby zpřesňovat. Všechna vyslaná energie byla v tomto modelu někde přijata.

6. Podle vlastností materiálů urči pro každý polygon, který podle našeho modelu energii přijal, kolik jí odrazí zpět do scény. Pokud je cílem aproximovat i pokročilé optické jevy, je nutné už při vybírání vzorků zaznamenávat směry příchozích nebo odražených paprsků a později je vhodně využít, já se zde ovšem budu zabývat pouze difusním odrazem, takže odraženou energií rozumím všesměrově odraženou.

7. Podle celkového množství odražené energie stanov vhodné nové N. Pokud například scéna odráží čtvrtinu přijaté energie, vhodné nové N může být čtvrtinové.

8. Dokud není N blízké nule, považuj zasažené polygony odrážející energii za nové primární zářiče, staré zářiče za obyčejné polygony a pokračuj bodem 2. Seznam starých primárních zářičů, ze kterých už byla střílena energie, a vygenerovaných stínových map udržuj stranou.

9. Pro každou stínovou mapu vygenerovanou během celého chodu algoritmu vygeneruj výsledný obraz s ostrými stíny.

10. Za výsledek považuj vážený průměr všech těchto obrazů, přičemž váhy jsou energie přiřazené mapám, resp. bodovým zdrojům, podle kterých byly mapy generovány.

Jde tedy o klasický výpočet primárního osvětlení aproximací všech plošných zářičů bodovými zdroji, metoda je ovšem efektivně rozšířena o získávání konfiguračních faktorů a výpočet všech významných odrazů od osvětlených ploch, tedy ke globálnímu osvětlení.

5.4 Přesnější řešení

Aproximace plošného zářiče bodovými zdroji může v jednodušším a rychlejším případě vypadat jako zhruba rovnoměrné rozmístění bodů. Ve většině

případů to pro interaktivní manipulaci se scénou s alespoň přibližným globálním osvětlením postačí. Pokud ale požadujeme přesnější řešení difusních odrazů, musíme ošetřit případ, kdy je osvětlena jen část polygonu, ale její osvětlení a odrazivost jsou přesto natolik významné, že se polygon stane zářičem. Pak je žádoucí umístit bodové zdroje pouze do osvětlené části polygonu. Jak ale efektivně vybírat pouze osvětlené body? Dosáhnout toho lze snadno, pokud budeme mít pro každý podle našeho modelu osvětlený polygon k dispozici seznam dostatečného množství dostatečně rozptýlených osvětlených bodů. Právě takový seznam ovšem můžeme sestavit z vybíraných vzorků ze stínových map. Máme pak zaručeno, že kdykoliv se osvětlený polygon stane zářičem, je to proto, že byl mnohokrát zasažen výběrem vzorku a v seznamu má tedy dostatek osvětlených bodů.

5.5 Kompenzace nevystřílené energie

Jak je z algoritmu patrné, počáteční volbou N (počet bodových zdrojů) stanovujeme i určitou mez, za kterou už odrazy světla zanedbáváme. V praxi je tato mez poměrně vysoká, odrazy už typicky nevytvářejí stíny a jiné pro pozorovatele významné jevy, jejich počítání je přitom ale nejpomalejší. Proto v případě, kdy je požadována hlavně vysoká rychlost a mnoho energie zůstane zanedbáno a neodraženo, je vhodné úbytek nějak kompenzovat. Jeden způsob je odhadnout kolik energie nestihneme odrazit a takové množství přidat zářičům. Jiná možnost je promítnout nevystřelenou energii do ambientního osvětlení, zvýšit osvětlení všem polygonům. Ani jedno není korektní, ale oboje s minimálními náklady přesnost výsledku zvyšuje. Nicméně ve zcela temných částech scény se pak může nesprávně objevovat ambientní osvětlení, případně chybět tam, kde díky odrazům má být. Po implementaci zde popisovaného algoritmu výpočtu globálního osvětlení se proto pokusím navrhnout lepší způsob využívající cachování.

5.6 Eliminace duplicitních map

Optimalizace pro scény s velmi významným sekundárním osvětlením (odrazy) je využít seznam polygonů-zářičů, ze kterých už byla energie vysílána (které už byly aproximovány bodovými zdroji a pro ně vygenerovány stínové mapy). Pokud později vznikne potřeba znovu ze zářiče vyslat energii, stačí

již existujícím mapám zvýšit množství vysílané energie a přírůstek hned rozdistribuovat do zářičem osvětlených polygonů (které už známe díky vybírání vzorků z map). Pokud je přírůstek energie dostatečně významný, můžeme aproximaci plošného zářiče bodovými zdroji rozšířit o další bodové zdroje. V reálných komplexních scénách, na které se zaměřuji, ovšem velmi výrazné sekundární osvětlení většinou není a tato optimalizace nepřinese žádný užitek.

5.7 Likvidace vysokých paměťových nároků

Dosud jsem při popisu algoritmu tiše předpokládal, že během generování snímku je dostatek paměti pro držení všech stínových map naráz, výsledný obraz je sestaven až v závěru a to jako vážený součet obrazů generovaných ze stínových map. Váhy přiřazené jednotlivým mapám (jejich bodovým zdrojům) se mohou průběžně měnit. Očekávání zatím nepodložené experimenty ale je, že v reálných komplexních scénách se váhy měnit prakticky nebudou. Je tedy možné navrhnout řešení s paměťovou složitostí konstantní vzhledem k počtu bodových zdrojů, tedy takové řešení, které bude za cenu malého snížení kvality držet v paměti vždy jen jednu stínovou mapu. Jeho princip je jednoduchý, mapy bude generovat postupně a hned je váženě přičítat k výsledku (v accumulation bufferu), žádné uchovávání nebude nutné. Jediné o co přijdeme je možnost měnit v průběhu výpočtu množství energie ve zdrojích. Nebude tedy možná předchozí optimalizace (eliminace duplicitních map), ale ta v reálných komplexních scénách velkou roli nehraje, zatímco minimalizací paměťových nároků odstraňujeme úzké hrdlo celého postupu. Takto by měl kvalitu výsledků limitovat už jen výkon 3D akcelerátoru a částečně procesoru.

5.8 Paralelizace

Algoritmus lze teoreticky dobře paralelizovat, každou sadu stínových map, obrazů s ostrými stíny a jejich vážený průměr může počítat jiný akcelerátor, vzorky z různých sad stínových map též mohou vybírat různé procesory. Poslední krok, zprůměrovat do výsledného obrazu meziprodukty vzniklé z každé sady, by si žádalo přesuny značných objemů dat po sběrnicích, ovšem i to lze navrhnout tak, aby rozšiřování systému o další akcelerátory a procesory přinášelo jen malou režii. V případě potřeby tedy lze sestavit paralelní systém schopný vizualizovat i mimořádně komplexní dynamické scény.

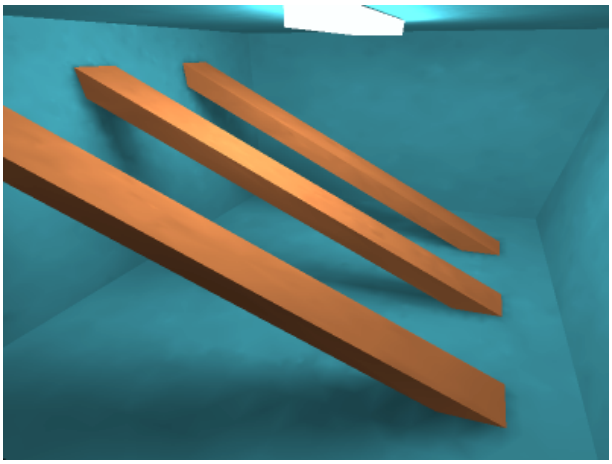
5.9 Kreslení jen viditelných polygonů

Při kreslení scén s ostrými stíny, ze kterých posléze složíme jeden snímek, jde vždy o tentýž záběr kamery, pouze scéna je vždy jinak osvětlená. Není tedy nutné věnovat velkou pozornost tomu, jak vykreslovat jen polygony viditelné a nezdržovat se těmi skrytými za bližšími objekty nebo mimo záběr kamery. Stačí celou scénu vykreslit jednou, otestovat z kterých polygonů je vidět alespoň jeden pixel a poté kreslit už jen ty. Zrychlení v komplexních scénách je značné, protože počet viditelných polygonů typicky nepřekračuje pár procent z celého počtu. Tuto triviální optimalizaci vykreslování zmiňuji proto, že v dalším odstavci budu využívat seznam v daném záběru viditelných polygonů a případně i kolik pixelů z nich je vidět. Získat všechny tyto údaje musíme už zde, takže později jsou k dispozici a není potřeba žádná práce navíc.

5.10 Cachování v komplexních scénách

V komplexních scénách s mnoha zdroji světla bude běžně nastávat situace, kdy naprostá většina zdrojů nijak neovlivňuje současný snímek, například při procházení domem to budou světla v ostatních místnostech. Ve speciálních případech je možné zřídit jakousi vyšší instanci, orákulum, které bude vždy vědět které zdroje lze v kterém bodě scény bez újmy ignorovat. Například v systému umožňujícím procházení domem a manipulaci s nábytkem (ale ne zdmi) by šlo o jakési rozdělení na sektory, orákulum by za významné označovalo zdroje v současném a případně nejbližších sektorech. Takto lze problém velkého množství nevýznamných zdrojů světla někdy vyřešit, ale co když máme umožnit libovolnou manipulaci s geometrií? Pak by bylo nesmírně obtížné, ne-li nemožné, orákulum implementovat. Proto jsem navrhl odlišný přístup pracující v obecně se měnících scénách. Využívá toho, že v reálných situacích se nebude neustále měnit vše, ale jen část geometrie, a bude tedy možné cachovat a znovu využívat údaje o viditelnosti v oblastech nevýznamných pro současný snímek. Tyto údaje budou periodicky, ale ne pro každý snímek, aktualizovány. Pokud zjistíme, že nějaký dosud nevýznamný zdroj světla začíná mít v současném záběru kamery význam, přejdeme na aktualizaci v každém snímku.

Jak tedy cachování funguje? Pro každý polygon-zářič, který aproximujeme bodovými zdroji (a generujeme pro ně stínové mapy z nichž vybíráním vzorků určujeme konfigurační faktory z polygonu do okolní scény), budeme



Obrázek 5.5: Ilustrační scéna - „zářivka“.

jeho konfigurační faktory cachovat. Z údajů jak moc zářič osvětluje jak moc viditelné polygony je jednoduché odvodit jeho význam pro současný záběr. Podle významu zářiče a dalších údajů (jeho plochy a výkonu) pak určíme kolika bodovými zdroji bude aproximován. Zatím ovšem známe jen význam přímého osvětlení, odrazy nejsou brány v potaz. I ty ovšem mohou hrát roli, proto je potřeba u polygonů-zářičů sledovat i jak významně osvětlují polygony ne přímo viditelné, ale pro osvětlení významné.

Výsledkem tedy je, že máme pro všechny potenciální zářiče určen jejich význam pro současný záběr. V komplexní scéně s mnoha zářiči budeme bodovými zdroji aproximovat typicky jen jejich malou část. I ostatní málo významné a zcela nevýznamné ovšem budeme s určitým větším intervalem pravidelně testovat (buď generovat stínovou mapu a vzorkováním určovat orientační konfigurační faktory nebo ze zdroje střílet paprsky).

Tímto přístupem jsme mimo jiné ošetřili případ scén kompletně osvětlených odraženým světlem. Pro skryté silné zářiče (například žárovka za stínítkem) budeme mít v cache konfigurační faktory a to poměrně přesné (pro zářiče vysílající velké množství energie vzorkujeme jemněji), nebudeme tedy muset v každém snímku znovu počítat co a jak žárovka osvětluje, zato budeme díky konfiguračním faktorům ze žárovky znát osvětlení okolních polygonů a pokud ty už budou přímo ozařovat viditelné polygony, budeme je považovat za zářiče a je aproximovat bodovými zdroji.

5.11 Stav implementace

Vzhledem k již dříve uvedeným okolnostem nebylo možné stihnout do doby odevzdání diplomové práce navrhovaný algoritmus implementovat. Po prvním měsíci práce jsou stále rozpracované indexové stínové mapy, pracuji na odstranění řady artefaktů (v případě neúspěchu lze vždy přejít k hloubkovým stínovým mapám aniž by bylo třeba ve zbytku programu cokoliv měnit). Funguje výpočet primárního osvětlení plošnými zářiči, přesněji jeho aproximace zadaným počtem bodových zdrojů. Na vstupu mohou být pouze statické scény. Testovací program má interface vhodný pouze pro ověření zda indexové stínové mapy a výpočet primárního osvětlení fungují a jak rychle.

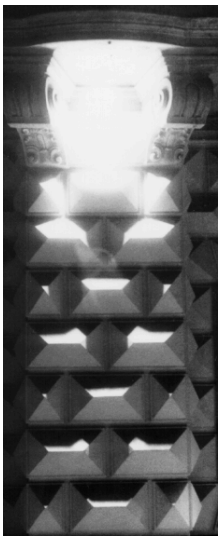
S dotažením systému k praktické demonstraci globálního osvětlení v komplexních dynamických scénách počítám na podzim 2001.

5.12 Shrnutí

Představil jsem algoritmus výpočtu globálního osvětlení (pouze difusní odrazy) v obecných dynamických scénách s minimálními paměťovými nároky a časovou složitostí odpovídající přibližně K vykreslením scény, kde K je počet zdrojů světla významně ovlivňujících současný záběr, přičemž K lze na úkor kvality snižovat až na *1*. S exponenciálním růstem výkonu 3D akcelerátorů tedy exponenciálně roste i složitost dynamických scén, na jejichž osvětlení můžeme nahlížet, další téměř lineární zrychlování je možné s paralelní architekturou obsahující řadu procesorů a 3D akcelerátorů. Pro případnou implementaci využívající indexové stínové mapy jsem popsal vznikající artefakty a navrhl jak jejich výskyt omezit. Na rozdíl od jiných známých implementací stínových map závislých na proprietárních rozšířeních OpenGL navrhuji jak vystačit se standardním OpenGL.

6

Závěr

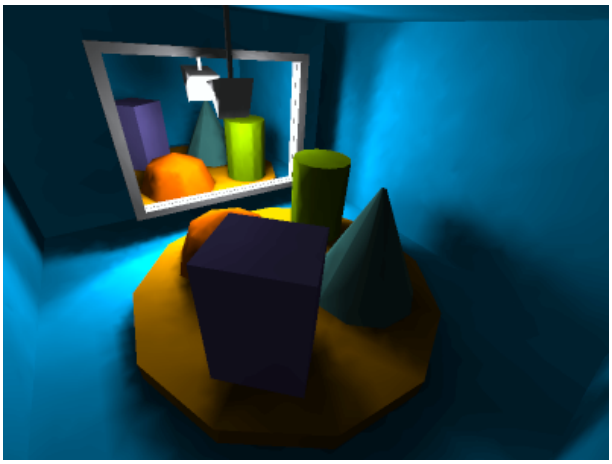


Z uměleckého hlediska se rozeznává osvětlení hlavní, doplňkové a efektové.

Malá československá encyklopedie

Vývoj nejen v oblasti počítačové grafiky je nikdy nekončící proces, a tak ani tuto diplomovou práci nelze chápat jako uzavřenou kapitolu, jde spíše o zachycení stavu po prvním roce vývoje. Během něj jsem

- Nastudoval, zařadil a zhodnotil desítky známých metod výpočtu osvětlení podle jejich vhodnosti pro komplexní dynamické scény (kapitola 2).
- Určil další techniky urychlující zobrazení a výpočet osvětlení, některé jako zcela nepostradatelné pro podporu skutečně komplexních scén, jiné alespoň jako významný zdroj zrychlení systému (kapitola 3).
- Navrhl a z velké části implementoval systém pro interaktivní manipulaci s dynamickými scénami s průběžně zpřesňovaným globálním osvětlením vycházející z Monte Carlo radiosity, využívající nově představený typ clusterů a rozšířený do dynamických scén (kapitola 4, shrnutí v 4.6).
- Představil způsoby jak implementovat indexové stínové mapy s OpenGL bez proprietárních rozšíření a jak potlačit nebo omezit artefakty z indexových stínových map známé, navrhl efektivní rozšíření stínových map ke globálnímu osvětlení vhodné i pro komplexní scény a zahájil implementaci výše uvedeného (kapitola 5, shrnutí v 5.12).

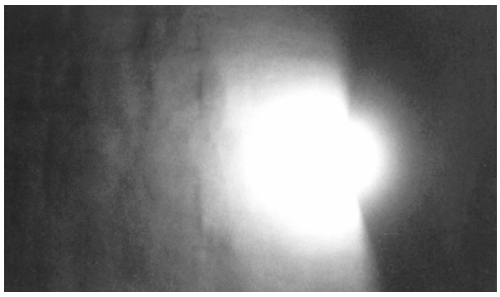


Obrázek 6.1: Ilustrační scéna - „zrcadlo“.

Celá práce a tedy i obě navrhovaná řešení se orientují na komplexní dynamické scény, ve kterých se počítat a aktualizovat globální osvětlení při zachování interaktivity dosud nikomu nepovedlo.

První implementovaný systém byl nasazen v praxi a prokázal svou schopnost vypořádat se s malými scénami. Ještě před testováním v komplexních scénách jsem ovšem navrhl perspektivnější řešení plně využívající výkonu 3D akcelerátorů, nabízející výsledky nezatížené šumem a tedy přijatelnější pro většinu potenciálních aplikací. Tento druhý systém jsem prezentoval pouze jako návrh, zahajuji ovšem jeho implementaci s cílem prakticky demonstrovat, že interaktivní práce s globálním osvětlením i v komplexních dynamických scénách možná je.

A



Fotodokumentace

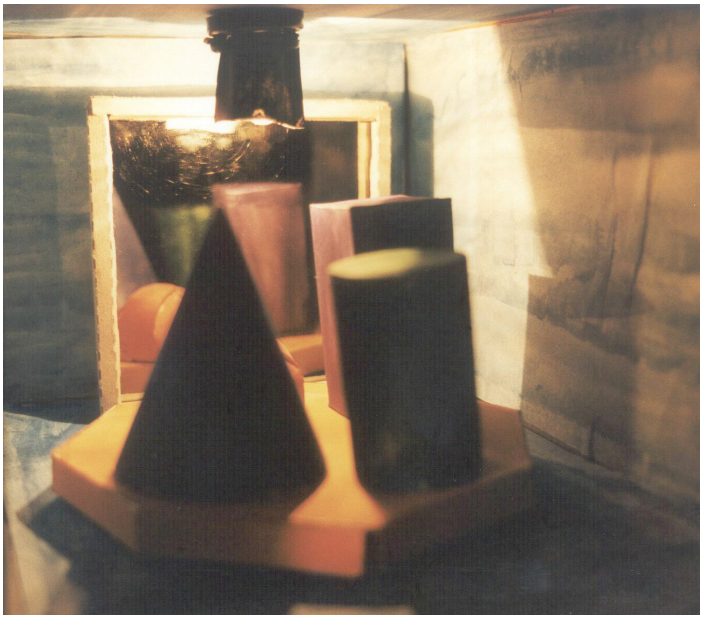
Vývoj nesměrových radiačních metod i raytracingu je dnes již prakticky ukončen, obě metody jsou na vrcholu svých možností a budoucnost patří (alespoň dle mého názoru) právě směrovým metodám.

Radim Halíř, diplomová práce, 1995

Obrázek A.1: Fotodokumentace - studium radiačních metod.

Obrázek A.2: Fotodokumentace - srovnávání s referenčními scénami.

Obrázek A.3: Fotodokumentace - Cornell box.



Obrázek A.4: Fotodokumentace - Zátiší se zrcadlem.

Obrázek A.5: Fotodokumentace - přípitek na další výzkum.

Literatura

[Ath78] P. Atherton, K. Weiler, D. Greenberg: *Polygon Shadows Generation*, Siggraph 78 Proceedings, 275–281

[Bau86] D. R. Baum, J. R. Wallace, M. F. Cohen, D. P. Greenberg: *The back-buffer algorithm: an extension of the radiosity method to dynamic environments*, The Visual Computer, 2:298–306

[Bek98] P. Bekaert, L. Neumann, A. Neumann, M. Sbert, Y. Willems: *Hierarchical Monte Carlo Radiosity*, Eurographics 98 Proceedings, 259–268

[Bek99] P. Bekaert, R. Cools, Y. D. Willems: *An Empirical Comparison of Monte Carlo Radiosity Algorithms*

[Ber86] P. Bergon: *A general version of Crow's shadow volumes*, IEEE Computer Graphics and Applications, September 1986, 17–28

[Bli88] J. F. Blinn: *Jim Blinn's corner: Me and my (fake) shadow*, IEEE Computer Graphics and Applications, January 1988, 82–86

[Bra00] S. Brabec, W. Heidrich, H. Seidel: *OpenGL shadow maps*, Technical Report TR 2000-4-002, Max-Planck-Institute for Computer Science

[Bro84] L. S. Brotman, N. I. Badler: *Generating soft shadows with a depth buffer algorithm*, IEEE Computer Graphics and Applications, October 1984, 71–81

[Cam91] A. T. Campbell: *Modelling Global Diffuse Illumination for Image Synthesis*, PhD thesis, Department of Computer Science, University of Texas at Austin

[Che90] S. E. Chen: *Incremental radiosity: An extension of progressive radiosity to an interactive image synthesis system*, Computer Graphics, 24(4), 135–144

[Chi92] N. Chin, S. Feiner: *Fast object-precision shadow generation for area light sources using BSP trees*, ACM Computer Graphics (Symp. on Interactive 3D Graphics), 21–30

[Chr97] Y. Chrysanthou, M. Slater: *Incremental Updates to Scenes Illuminated by Area Light Sources*, Rendering Techniques 97 (Proc. of Euro-Graphics Workshop on Rendering), strany 103-114

[Coh86] M. F. Cohen, D. P. Greenberg, D. S. Immel, P. J. Brock: *An Efficient Radiosity Approach for Realistic Image Synthesis*, IEEE Computer Graphics and Applications, March 1986, 26–35

[Coh88] M. F. Cohen, S. E. Chen, J. R. Wallace, D. P. Greenberg: *A Pogressive Refinement Approach to Fast Radiosity Image Generation*, Computer Graphics (Siggraph 88 Proceedings), August 1988, 75–84

[Coh93] M. F. Cohen, J. R. Wallace: *Radiosity and Realistic Image Synthesis*, Academic Press, Inc. Cambridge, MA 02 139, USA

[Cro77] F. C. Crow: *Shadow algorithms for computer graphics*, Computer Graphics (Siggraph 77 Procedings), 242–248

[Die94] P. J. Diefenbach, N. Badler: *Pipeline Rendering: Interactive refractions, reflections and shadows*, Displays: Special Issue on Interactive Computer Graphics

[Die96] P. J. Diefenbach: *Pipeline Rendering: Interaction and realism through hardware-based multi-pass rendering*, Dissertation, University of Pennsylvania

[Dre94] G. Drettakis, E. Fiume: *A fast shadow algorithm for area light sources using backprojection*, ACM Computer Graphics, July 1994, 223–230

[Dre97] G. Drettakis, F. Sillion: *Interactive Update of Global Illumination Using a Line-Space Hierarchy*, Siggraph 97 Proceedings

[Dur97] F. Durand, G. Drettakis, C. Puech: *The Visibility Skeleton: A Powerful And Efficient Multi-Purpose Global Visibility Tool*, Siggraph 97 Proceedings

[Dur99] F. Durand, G. Drettakis, C. Puech: *Fast and Accurate Hierarchical Radiosity Using Global Visibility*, ACM Transactions on Graphics, April 1999, vol 18, no 2, 128–170

[For94] D. Forsyth, C. Yang, K. Teo: *Efficient radiosity in dynamic environments*, Photorealistic Rendering Techniques, June 1994

[Fuch80] H. Fuchs: *On Visible Surface Generation by A Priori Tree Structure*, Computer Graphics (Siggraph 80), 124–133

[Fuch85] H. Fuchs, J. Goldfeather, J. P. Hultquist, S. Spach, J. D. Austin, F. P. Brooks, J. G. Eyles, J. Poulton: *Fast spheres, shadows, textures, transparencies and image enhancements in Pixel-Planes*, Computer Graphics (Siggraph 85 Proceedings), July 1985, 111–120

[Fuj86] A. Fujimoto, T. Tanaka, K. Iwata: *ARTS: Accelerated Ray Tracing System*, IEEE Computer Graphics and Applications, 6:4, 16–26

[Geo90] D. W. George, F. Sillion, D. P. Greenberg: *Radiosity Redistribution for Dynamic Environments*, IEEE Computer Graphics and Applications, 10(4)

[Gla84] A. S. Glassner: *Space Subdivision for Fast Ray Tracing*, IEEE Computer Graphics and Applications, 4:10, 15–22

[Gor84] C. M. Goral, K. E. Torrance, D. P. Greenberg, B. Battaile: *Modeling the interaction of light between diffuse surfaces*, Computer Graphics, July 1984, 213–222

[Gor93] S. J. Gortler, P. Schroder, M. F. Cohen, P. Hanrahan: *Wavelet Radiosity*, Computer Graphics 93 Proceedings, 221–230

[Gra00] X. Granier, G. Drettakis, B. Walter: *Fast Global Illumination Including Specular Effects*, Eurographics 2000

[Hae90] P. Haeberli, K. Akeley: *The accumulation buffer: Hardware support for high-quality rendering*, Computer Graphics (Siggraph 90 Proceedings), August 1990, 309–318

[Hal93] R. Halíř: *Směrový přístup v radiačních metodách*, Diplomová práce, MFF UK

[Han90] P. Hanrahan, D. Salzman: *A Rapid Hierarchical Radiosity Algorithm for Unoccluded Environments*, Princeton University, Depatrment of Computer Science, Technical Report CS-TR-281-90

[Han91] P. Hanrahan, D. Salzman, L. Aupperle: *A Rapid Hierarchical Radiosity Algorithm*, Computer Graphics (Siggraph 91 Proceedings), 25:4, 197–206

[Has99] J. Hasenfratz, C. Damez, F. Sillion, G. Drettakis: *A Practical Analysis of Clustering Strategies for Hierarchical Radiosity*, Eurographics 99

[Hec92a] P. Heckbert: *Radiosity in flatland*, Computer Graphics Forum (Eurographics 92), 181–192

[Hec92b] P. Heckbert: *Discontinuity Meshing for Radiosity*, Eurographics Rendering Workshop, May 1992, 203–216

[Hec97a] P. Heckbert. M. Garland: *Survey of Surface Simplification Algorithms*, SIGGRAPH 97

[Hec97b] P. Heckbert, M. Herf: *Simulating soft shadows with graphics hardware*, Technical Report CMU-CS-97-104, Carnegie Mellon University

[Hei91] T. Heidmann: *Real shadows, real time*, Iris Universe, 28–31, Silicon Graphics, Inc.

[Hei99] W. Heidrich, H. Seidel: *Realistic, Hardware-accelerated Shading and Lighting*, Max-Planck-Institute for Computer Science

[Hei00] W. Heidrich, S. Brabec, H. Seidel: *Soft Shadow Maps for Linear Lights*, Max-Planck-Institute for Computer Science

[Her97] M. Herf: *Efficient Generation of Soft Shadow Textures*, Technical Report CMU-CS-97-138, Carnegie Mellon University

[Kay86] T. L. Kay, J. T. Kajiya: *Ray Tracing Complex Scenes*, Siggraph 86 Proceedings, 269–278

[Kel96] A. Keller: *Quasi-Monte Carlo Radiosity*, Eurographics Rendering Workshop 96, 101–110

[Lis92] D. Lischinski, F. Tampieri, D. P. Greenberg: *Discontinuity meshing for accurate radiosity*, IEEE Computer Graphics & Applications, 12(6), 25–39

[Lis93] D. Lischinski, F. Tampieri, D. P. Greenberg: *Combining hierarchical radiosity and discontinuity meshing*, ACM Computer Graphics, volume 27, 199-208

[Los97] C. Loscos, G. Drettakis: *Interactive High-Quality Soft Shadows in Scenes with Moving Objects*, Computer Graphics Forum Eurographics 97, 219–230

[Mul94] S. Muller, F. Schoffel: *Fast radiosity repropagation for interactive virtual environments using a shadow-form-factor-list*, Photorealistic Rendering Techniques, June 1994

[Nis83] T. Nishita, E Nakamae: *Half-tone representation of 3D objects illuminated by area sources or polyhedron sources*, COMSPAC 83 Proceedings, 237–242

[Nis85] T. Nishita, E. Nakamae: *Continuous Tone Representation of Three-Dimensional Objects Taking Account of Shadows and Interreflection*, Computer Graphics (Siggraph 85 Proceedings), 19:3, 23–30

[Ort96] R. Orti, S. Riviere, F. Durand, C. Puech: *Radiosity for Dynamic Scenes in Flatland with the Visibility Complex, Computer Graphics Forum Eurographics 96*, 237–248

[Pop00] J. Pope, A. Chalmers: *Improving Hierarchical Monte Carlo Radiosity Algorithms*

[Ree87] W. T. Reeves, D. H. Salesin, R. L. Cook: *Rendering antialiased shadows with depth maps*, Computer Graphics (Siggraph 87 Proceedings), July 1987, 283–291

[Sch96] G. Schaufler, W. Sturzlinger: *A Three Dimensional Image Cache for Virtual Reality*, Computer Graphics Forum Eurographics 96, 227–235

[Sch93] P. Schroder, P. Hanrahan: *A Closef Form Expression for the Form Factor Between Two Polygons*, Princeton University, Department of Computer Science, Technical Report CS-404-93

[Seg92] M. Segal, C. Korobkin, R. Widenfelt, J. Foran, P. Haeberli: *Fast shadow and lighting effects using texture mapping*, Computer Graphics (Siggraph 92 Proceedings), July 1992, 249–252

[Seg99] M. Segal, K. Akeley: *The OpenGL Graphics System: A Specification (Version 1.2.1)*, Silicon Graphics, Inc.

[Sha94] E. Shaw: *Hierarchical radiosity for dynamic environments*, master's thesis, Cornell University

[Sil94] F. Sillion: *Clustering and volume scattering for hierarchical radiosity calculations*, Eurographics 94 Proceedings

[Sil95] F. Sillion, G. Drettakis: *Feature-based Control of Visibility Error: A Multi-resolution Clustering Algorithm for Global Illumination*, Siggraph 95, Computer Graphics Proceedings, 145–152

[Smi94] B. Smits, J. Arvo, D. P. Greenberg: *A clustering algorithm for radiosity in complex environments*, Siggraph 94 Proceedings, 435–442

[Ste94] A. J. Stewart, S. Ghali: *Fast computation of shadow boundaries using spatial coherence and backprojections*, ACM Computer Graphics, July 1994, 231–238

[Sud96] O. Sudarsky, C. Gotsman: *Output-Sensitive Visibility Algorithms for Dynamic Scenes with Applications to Virtual Reality*, Computer Graphics Forum Eurographics 96, 249–258

[Tan97] T. Tanaka, T. Takahashi: *Fast Analytic Shading and Shadowing for Area Light Sources*, Computer Graphics Forum Eurographics 97, 231–240

[Tob97] R. F. Tobler, A. Wilke, M. Feda, W. Purgathofer: *A Hierarchical Subdivision Algorithm for Stochastic Radiosity Methods*, Eurographics 97 Proceedings, 193–203

[Ude99] T. Udeshi, C. Hansen: *Towards interactive, photorealistic rendering of indoor scenes: A hybrid approach*, Rendering Techniques 99 (Eurographics Rendering Workshop 99 Proceedings), 63–76

[Wal99] B. Walter, G. Drettakis, S. Parker: *Interactive Rendering Using the Render Cache*, Rendering Techniques 99

[Wil78] L. Williams: *Casting curved shadows on curved surfaces*, Siggraph 78 Proceedings, 270–274

[Wil99] A. J. Willmot, P. Heckbert, M. Garland: *Face Cluster Radiosity*, Eurographics 99

[Woo92] A. Woo: *Graphics Gems III, kapitola The Shadow Depth Map Revisited*, Academic Press, 338–342

[Wor95] A. Worrall, C. Willis, D. Paddon: *Dynamic Discontinuities for Radiosity*, Edugraphics + Compugraphics Proceedings, 367–375

[Wor98] A. Worrall, D. Hedley, D. Paddon: *Interactive Animation of Soft Shadows*

[Zat93] Zatz, Harold: *Galerkin Radiosity: A Higher Order Solution for Global Illumination*, Siggraph 93 Conference Proceedings, 213–220