VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

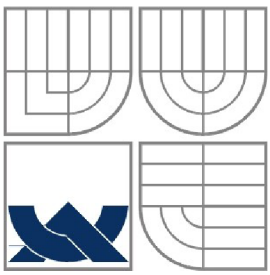
DISTRIBUOVANÉ SLEDOVÁNÍ PAPRSKU

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

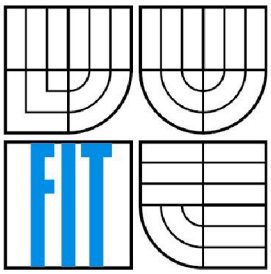
AUTOR PRÁCE
AUTHOR

RADEK SLOVÁK

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DISTRIBUOVANÉ SLEDOVÁNÍ PAPRSKU

DISTRIBUTED RAYTRAYCING

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

RADEK SLOVÁK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. ADAM HEROUT PH.D.

BRNO 2008

Abstrakt

Práce se zabývá studiem globální zobrazovací metody distribuované sledování paprsku. Ta z matematického popisu scény generuje dvourozměrné obrazy o vysoké kvalitě a realističnosti. Práce rozebírá problematiku a vysvětluje postupy řešení s touto technikou spojené. Popisuje také neoddělitelnou součást této metody a to klasické sledování paprsku.

Klíčová slova

distribuované sledování paprsku, matné povrchy, průsvitné materiály, měkké stíny, hloubka ostrosti, rozmazáni obrazu pohybem, interpolace, zrcadlové odrazy, lom světla, Phongův osvětlovací model, sledování paprsku

Abstract

This project describes method for global display of distributed ray tracing. The method generates high quality and realistic two-dimensional images from mathematical specification of scene. The methods for solution of this issue are presented and analyzed. Integral part of this method - conventional ray tracing - is also presented.

Keywords

distributed ray tracing, diffuse reflection, diffuse refraction, soft shadows, depth of field, motion blur, interpolation, mirror reflection, refraction, Phong shading model, ray tracing

Citace

Radek Slovák: Distribuované sledování paprsku, bakalářská práce, Brno, FIT VUT v Brně, 2008

Distribuované sledování paprsku

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Adama Herouta, Ph..D.
Další informace mi poskytli Bc. Jan Pinter a Michael Hrabánek.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Radek Slovák
19.5.2008

Poděkování

Děkuji Ing. Adamovi Heroutovi, Ph..D., který mi jako vedoucí tohoto projektu byl ochoten věnovat čas a konzultace a především za poskytnuté cenné informace k projektu. Dále bych rád poděkoval Janu Pinterovi za cenné informace a nápady při vývoji programu a Michaeli Hrabánkovi za je pomoc s překladem několika zdrojů.

Obsah

1 Úvod

V počítačové grafice je sledování paprsku (angl. **ray tracing**) metoda sloužící v podobě programu, který nazýváme **ray tracer**, ke generování velmi realistických, či naopak nereálných 2D zobrazení 3D scény z matematického popisu, kdy se snímají jednotlivé paprsky světla dopadající na projekční plochu a vytvářejí tak velmi reálně vypadající obraz.

Tato metoda má velmi staré kořeny, které sahají až do starověkého Řecka. Tam lidé věřili, že obraz, který vidí, byl získáván detekčními paprsky vysílanými z očí, které snímaly okolní svět. Samozřejmě to pravda nebyla, ale nelze upřít podobu s dnešním ray tracingem. První praktické využití ray tracingu se objevilo až na přelomu 15. a 16. století, kdy Albrecht Dürer použil strunu k tomu, aby přesně zobrazil 3D objekt na 2D plátno při zachování perspektivy.

V roce 1968, Arthur Appel, pracující ve společnosti IBM [1], poprvé navrhl a implementoval ray tracing do programu. Byl to dokonce první implementovaný algoritmus řešící zpracování globálního osvětlení s interakcemi mezi objekty. Bohužel metoda nereflektovala nedokonalé povrchy reálných objektů a měkké stíny z rozměrných světel. V roce 1984 Robert L. Cook vydal publikaci [2] popisující **distribuované sledování paprsku** (angl. Distributed ray tracing), které tyto nedostatky odstranilo a přidalo spoustu dalších simulovaných vlastností (hloubka ostrosti, rozmazání pohybem). Ovšem za cenu několikanásobně větších nároků na výkon, než u klasické metody.

Vzhledem k tomu, že ray tracing částečně simuluje reálné vlastnosti světla, má velmi široké využití. Nejviditelnější je ve filmovém průmyslu při renderování scén jako jsou např. bitva mezi desetitisíci vojáky. Velké využití nachází také v architektuře, ve strojírenství, medicíně a podobně.

Techniku lze zařadit do obrazových vizualizačních metod, konkrétně do globálních až komplexních, v závislosti na míře implementace fyzikálních zákonů.

Cílem mé bakalářské práce je popsat, navrhnout a implementovat techniku distribuovaného sledování paprsku do programu, předvést výsledky v podobě obrazových výstupů z programu a porovnat je vůči klasickému ray tracingu. Dále zhodnotit kvalitu výstupu a časovou náročnost výpočtů.

V následujících kapitolách vás nejdříve provedu základy klasického ray tracingu, které jsou nedílnou součástí distribuovaného ray tracingu, poté rozšířeného o stochastické prvky v podobě distribuovaného ray tracingu. Následuje popis zpracování jednotlivých efektů, návrh a implementace ray traceru a na závěr uvedu výsledky snažení, jeho výhody, nevýhody a jeho možná rozšíření v budoucnosti.

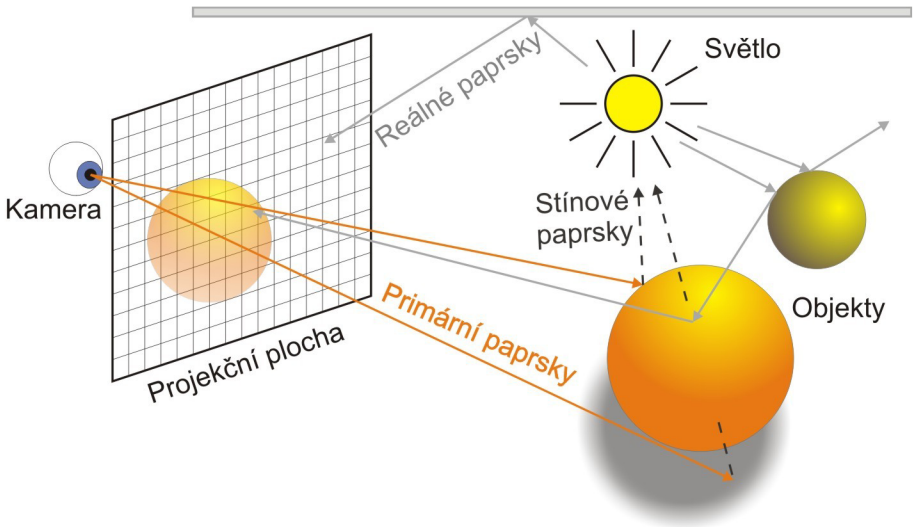
2 Sledování paprsku

To, co vidíme v reálném světě, nazýváme obraz, či zobrazení. Lze jej popsat různými způsoby abstrakce podle úrovně znalostí. Člověk může například říci, že vidí auto stojící na zemi, nebo také světlo ozařující okolní objekty. Pokud velmi dobře rozumí fyzice, může prohlásit, že vidí velké množství proudících fotonů, světelné paprsky odrážející se od okolních předmětů a také elektromagnetické záření, které má takovou vlnovou délku, že je oči vnímají jako barvy.

Tři posledně zmíněné druhy znalostí představují fyzikální základ toho, co vidíme – světlo šířící se v prostředí. Právě těchto fyzikálních zákonů se v počítačové grafice zaměřené na realistické zobrazování hojně využívá. Konkrétně ve zobrazování založeném na ray tracingu se nejvíce užívá geometrické optiky v podobě simulace odrážení paprsků.

Cílem techniky ray tracingu je získat výstup v podobě dvourozměrného pole barevných bodů, (pixelů) z 3D scény složené z objektů a světel o různých vlastnostech, pomocí simulace šíření paprsků scénou.

Ve skutečnosti, podle geometrické optiky, světla produkují světelné paprsky, ty narážejí v prostředí do objektů, od kterých se odrážejí a lámou. Některé z nich nakonec dopadnou na čočku oka, přes kterou se lomí do sítnice. Tyto dopadlé paprsky jsou pak vnímány jako barevné světlo a utváří tak obraz, který vidíme.



Ilustrace 1: Reálné chování světla (šedou) a ray tracing (oranžovou a černou)

V počítačové grafice funguje ray tracer na podobném principu s tím, že se dělí na 2 druhy, podle toho, jakou technikou je implementován:

- **sledování paprsku (**anglicky **ray tracing**, **foward ray tracing) -** Paprsky se vrhají (angl. ray casting) ze světla, šíří se prostředím a jen velmi malá část z nich dopadne na projekční plochu, což dělá tuto metodu velmi neefektivní.
- **Zpětné sledování paprsku (**anglicky **back ray tracing)** – Paprsky se nevrhají ze světel, ale z kamery přes projekční plochu a sleduje se, zda paprsky ve scéně narazí na světlo. V této metodě se využívá faktu, že paprsky se šíří od světla do projekční plochy a zpět po totožné trase (při zanedbání některých zákonů fyziky). Díky tomu náročnost na výkon několikanásobně klesne, protože se vrhají a počítají jen paprsky které mají vliv na výsledný obraz. Tuto metodu jsem využil při implementaci programu.

Jelikož že se foward ray tracing v počítačové grafice téměř neužívá, nazýváme pro zjednodušení back ray tracing jen jako **ray tracing**. Toto označení budu dále využívat v textu.

V dalších kapitolách proberu jednotlivé části ray tracingu od základních součástí, přes jejich interakci a algoritmy až po použitý osvětlovací model.

2.1 Použité pojmy

Jen ve zkratce vysvětlím základní pojmy, které jsou v technice sledování paprsku a jeho modelu šíření světla použity. Pojmy, které jsou velmi důležité a provádějí se s nimi hlavní výpočty.

2.1.1 Vektory a body

Vektory použité v textu jsou míněné jako trojrozměrné. Bod je reprezentován jako 3D vektor od počátku globálního souřadného systému.

2.1.2 Barva

Barva je reprezentací toho co naše oko vnímá jako různě intenzivní barevné světlo. Pro účel ray tracingu bude stačit složení ze tří složek: Červená, zelená a modrá (RGB), což se dá označit za trojrozměrný vektor. Jednotlivé složky jsou určeny reálným číslem o velmi velkém rozsahu (v C++ double). Při ukládání do souboru, se barvy redukují na rozsah 8 bitů (0–255) pro každou složku, tedy každý bod obrazu je o hloubce 24 bitů (přibližně 16 miliónů barev).

2.1.3 Projekční plocha

Projekční plocha reprezentuje to co vidí kamera, tedy digitální obraz z výstupu programu. Je to soubor pixelů o různých barvách. Projekční plocha má určitou šířku a výšku a z nich odvozený poměr stran.

2.1.4 Objekt

Objekt je reprezentací viditelného předmětu ve scéně. Každý objekt je matematicky popsaný tvar, který je určen polohou v čase, rotací a hybnou silou, kterou disponuje. Dále je ke každému objektu přiřazen materiál určující barvu a chování paprsků na povrchu objektu (např. odrazivost, lesklost, barva atp.).

2.1.5 Scéna

Scéna je soubor všech objektů. Zajišťuje jejich správu a hledá nejbližší objekt, který protíná cestu paprsku a vrací vlastnosti tohoto průsečíku.

2.1.6 Kamera

Kamera je reprezentací oka, případně kamery v prostoru. Kamera určuje jaké vlastnosti bude mít projekční plocha potažmo výstupní obraz. Mezi tyto vlastnosti patří: Pozice projekční plochy, její natočení ve 3D, úhel pohledu (výhledu) kamery a hloubka ostrosti obrazu (v distribuovaném ray tracingu). Dále také pozice a otočení kamery v čase a také to který čas v simulované scéně bude kamera zabírat. Podrobněji kameru popíši ve 4. kapitole, věnující se podrobněji implementovaným součástem, kde ukážu jak se propočítávají jednotlivé parametry.

2.1.7 Světlo

Bodový zdroj světla je základní, v realitě se nevyskytující typ osvětlení vyzařující světlo z nekonečně malého bodu Díky tomu vrhá přes objekty velmi nereálné přesně ohraničené stíny. Jeho jedinými vlastnostmi jsou pozice a barva.
V distribuovaném ray tracingu se dále vyskytují rozměrné světla. Tyto světla mají navíc od bodového světla také tvar. Objekty osvětlené tímto typem světla mají pak měkké stíny, tak jak je známe ze skutečnosti.

2.2 Paprsek

Paprsek je použit jako abstrakce šíření světla v podobě počátku a směru kterým se šíří světlo.

Z geometrické optiky je známo, že pokud se světlo šíří homogenním prostředím a dopadá na překážky dostatečně velké ve srovnání s vlnovou délkou světla, tak lze pozorovat, že se šíří přímočaře. Toto přímočaré šíření světla lze převést na svazek paprsků, který šíří světlo.

Dále jsou, z principu základního a distribuovaného sledování paprsku, zanedbány některé vlastnosti šíření světla, jako jsou disperze, ohyb, předávání energie v podobě vyzařování a podobně.

Paprsek lze zapsat matematicky jako polopřímku určenou počátečním bodem a vektorem určujícím její směr. Rovnice polopřímky r v parametrickém tvaru vypadá takto:

$$r: \vec{p} = \vec{o} + t \cdot \vec{d} \qquad t \geq 0$$

Vzorec 1: Vzorec pro výpočet bodu na polopřímce

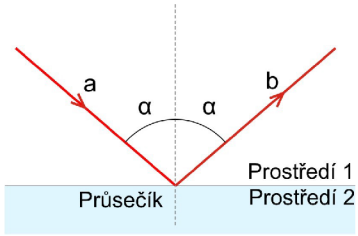
Kde $\vec{p}$ je výsledný bod, $\vec{o}$ počáteční bod paprsku, $\vec{d}$ směrový vektor paprsku a t parametr určující vzdálenost výsledného bodu od počátku polopřímky v případě, že $\vec{d}$ je normalizován na jednotkovou velikost.

Při výpočtech lze rozlišit paprsky do kategorií, které jsou následující:

- **Primární paprsky** jsou ty, které procházejí projekční plochou.
- **Sekundární paprsky** se odrážejí, nebo lámou ve scéně na povrchu objektů, ale nedopadají na projekční plochu.
- **Stínové paprsky** jsou pomocné paprsky, které slouží pro zjištění, jestli objekt leží v daném místě ve stínu, nebo na něj dopadá světlo. Podle toho se dále určuje výsledná barva.

2.2.1 Odraz paprsku

Odraz paprsku od odrazivého povrchu v ray traceru se řídí zákonem odrazu z vlnové optiky.



Ilustrace 2: Odraz paprsku

Platí, že úhel, který svírá příchozí paprsek s normálou povrchu, se rovná úhlu, který svírá odražený paprsek s normálou. Výpočet vektoru odraženého paprsku je velmi dobře nastíněn v [2]. Vzorec odraženého paprsku $\vec{r}$ je následující:

$$\vec{r} = \vec{e} - 2(\vec{e} \cdot \vec{n})\vec{n} \qquad |\vec{n}| = 1, |\vec{e}| = 1,$$

Vzorec 2: Vzorec pro výpočet odrazu paprsku

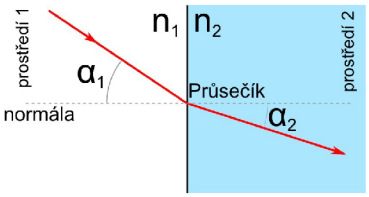
Kde $\vec{e}$ je vektor příchozího paprsku a $\vec{n}$ normálový vektor objektu v místě dopadu paprsku.

2.2.2 Refrakce paprsku

K refrakci (lomu) paprsků dochází u objektů z průhledného, či průsvitného materiálu, kdy část paprsků se lomí na tomto rozhraní dvou prostředí (např. vzduch - voda) a pokračují dále tělesem, do kterého narazily. Hlavními faktory při výpočtu lomeného paprsku je, pod jakým úhlem paprsek dopadá a jaké indexy lomu mají jednotlivé prostředí. Lom paprsku je popsán Snellovým zákonem:

$$\frac{\sin(\alpha_1)}{\sin(\alpha_2)} = \frac{n_1}{n_2}$$

Kde α_1 a α_2 jsou úhly příchozího a lomeného paprsku vůči normále v bodě průsečíku. n_1 a n_2 jsou indexy lomu pro původní a nové prostředí. Pro názornost uvádím obrázek.



Ilustrace 3: Lom paprsku

Odvození výpočtu vektoru lomeného paprsku je velmi důkladně popsáno v [3], proto jej zde nebudu uvádět. Uvedu ale konečný vzorec pro výpočet lomeného paprsku použitý v ray traceru:

$$\vec{t} = \frac{n_1}{n_2}\vec{e} - \left(\frac{n_1}{n_2}\cos(\alpha_1) + \sqrt{1-\sin^2(\alpha_2)}\right)\cdot\vec{n} \qquad \sin^2(\alpha_2) \le 1$$

$$\cos(\alpha_1) = \vec{e}\cdot\vec{n}, \qquad \sin^2(\alpha_2) = \left(\frac{n_1}{n_2}\right)^2\left(1-\cos^2(\alpha_1)\right) \qquad |\vec{n}| = 1,\ |\vec{e}| = 1$$

Vzorec 3: Vzorec pro výpočet lomeného paprsku

Kde $\vec{t}$ je výsledný vektor, $\vec{e}$ vektor příchozího paprsku a $\vec{n}$ normálový vektor směřující ven z objektu. V případě, že příchozí paprsek putuje tělesem a lomený vychází z jeho povrchu je nutné normálu otočit dovnitř objektu.

Při lámání paprsků je nutné podotknout, že každý materiál částečně absorbuje světlo a tím způsobuje útlum. Tento útlum roste se vzdáleností exponenciálně. Pro potřeby počítačové grafiky se velmi často linearizuje. V tomto ray traceru však útlum světla zanedbávám, protože jeho vliv na výstup je spíše zanedbatelný.

2.3 Výpočet průsečíku

Pokaždé, když je paprsek imaginárně vrhnut, zjišťuje se jestli protíná nějaký objekt. Pokud ano, tak se spočítá vzdálenost od kamery. Poté je již jednoduché zjistit polohu průsečíku z rovnice polopřímky

paprsku vycházejícího z kamery. Existují obecně 2 druhy řešení výpočtu průsečíku resp. vzdálenosti průsečíku od kamery:

- Algebraické
- Geometrické

Algebraické řešení má výhodu v tom, že jej lze vždy spočítat pomocí jisté metody výpočtů rovnic (substituční apod.). Geometrické řešení má výhodu v tom, že lze o něco dříve určit, jestli paprsek objekt protíná, nebo ne. Avšak celý výpočet včetně průsečíku, může naopak zabrat více času. V geometrickém řešení vidím, dle mého názoru jednu nevýhodu. Objekty o velmi složitých tvarech, jako například kvadrika budou mít velmi složité řešení, či nepůjdou geometricky, domnívám se, spočítat vůbec.

Jelikož je této tématice na internetu věnováno velký prostor, odkážu vás na tyto zdroje. Za jednu z nejlepších publikací považuji velmi obsáhlý zdroj [4], který shromažďuje odkazy na jednotlivá řešení průsečíků různých geometrických útvarů mezi sebou.

2.4 Algoritmus sledování paprsku

Metoda ray tracing se dá velmi přehledně popsat algoritmem v podobě zjednodušeného pseudokódu uvedeného níže:

```
Pro každý bod projekční plochy {
    Vytvoř paprsek s počátkem v kameře, procházející přes daný bod.
    Vypočti barvu pro paprsek (paprsek).
    Výslednou barvu zapiš do bodu v projekční ploše
}

Vypočti barvu pro paprsek (paprsek) {
    Nastav výsledný objekt na neznámý a výslednou barvu na černou.
    Pro každý objekt ve scéně {
        Jestliže paprsek protíná objekt, tak
            Pokud je vzdálenost objektu menší než nejbližší objekt
                Nastav tento objekt jako nejbližší a nastav vzdálenost
    }


    Jestliže je výsledný objekt známý, tak {
        Pro každé světlo {
            Vyšli stínový paprsek z průsečíku směřující ke světlu.
            Pokud cesta ke světlu není křížená jiným objektem, tak {
                Spočítej barvu pomocí osvětlovacího modelu pro dané světlo.
```

```
                Vrácenou barvu přičti k výsledné barvě.
            }
        }
        Pokud má objekt odrazivé vlastnosti, tak: {
            Generuj odražený paprsek.
            Vypočti barvu pro paprsek (odražený paprsek).
            Vrácenou barvu přičti k výsledné barvě.
        }
        Pokud má objekt průhledné vlastnosti, tak: {
            Generuj rekurzivní lomený paprsek.
            Vypočti barvu pro paprsek (lomený paprsek).
            Vrácenou barvu přičti k výsledné barvě.
        }
    }
    Vrať výslednou barvu.
}
```

Tento algoritmus je popisem, jak zmíněná metoda funguje. Nejdříve se tedy vyšle primární paprsek procházející projekční plochou, určí se který objekt protíná jako první, určí se průsečík. V tomto průsečíku se určí normálový vektor objektu. Vyšle se jediný stínový paprsek ke každému světlu. Pokud se nenachází překážka mezi bodem a světlem, tak se spočítá osvětlovací model pro daný bod. Poté se vyšlou rekurzivní odražený a lomený paprsek. Nakonec se jednotlivé barvy osvětlovacího modelu a rekurzivních paprsků sečtou s určitou váhou (podle materiálu) do výsledné barvy a ta se zapíše do daného bodu na projekční ploše.

Pro rekurzivní sekundární paprsky platí to samé, až na to že se výsledná barva nezapíše do projekční plochy, ale přičte se váhově k barvě počítané pro jiný sekundární či primární paprsek, v daném bodu ve scéně. A takto se vrací barva rekurzivně zpět až se vrátí primárnímu paprsku.

Stínové paprsky a barva od světel se nepočítají v rekurzi, ale v cyklu, protože se počítá barva v daném bodě pomocí zjednodušeného osvětlovacího modelu pokud není objekt ve stínu. Rekurzivní volání sekundárního lomeného a odraženého paprsku je nutné omezit maximálním hloubkou rekurze.

2.5 Osvětlovací model a materiál

Osvětlovací model určuje jak se bude ve scéně propočítávat světlo, které nakonec dorazí na projekční plochu. Existují 2 skupiny těchto modelů:

- Fyzikální
- Empirický

V reálném světě se veškeré světlo odráží a lomí, od překážek na které narazí. Přitom intenzita světla slábne a mění svoji barvu podle povrchu tělesa. Ovšem nejedná se jen o odrazy a lomy o povrch tělesa, ale také o několikanásobné vnitřní odrazy v nerovnostech povrchu. Tento reálný model šíření světla označujeme jako **fyzikální model.**

V počítačové grafice je vzhledem k náročnosti na výkon a paměť nereálné matematicky popisovat geometricky objekt i se všemi nerovnostmi a poté počítat obrovské množství rekurzivních odrazů a lomů (viz. Kapitola 6). A proto se zavedly různé zjednodušené osvětlovací modely, které vlastnosti povrchu těles popisují pomocí vzorce **empiricky**.

Mezi nejčastěji používané empirické modely (v ray tracingu) se slušnými výsledky při relativně nízké náročnosti patří model Bui-Tuong Phonga (zkráceně **Phongův model)** [10].

Všechny vlastnosti, které mají vliv na šíření světla o či přes překážky označujeme v ray tracingu souhrnně jako vlastnosti **materiálu**.

V této kapitole se zaměřím především na popis jednotlivých vlastností materiálu a výpočtu světla ve scéně pomocí Phongova osvětlovacího modelu. Okrajově uvedu mapování textur.

2.5.1 Vlastnosti materiálu

Předtím, než uvedu výpočty s Phongovým modelem, seznámím vás s vlastnostmi materiálu použité v tomto modelu s rozšířením pro výpočet distribuovaných výpočtů. Patří mezi ně:

- **Ambient(angl.)** (česky by se dalo přeložit jako koeficient působení okolního rozptýleného světla). Tato vlastnost udává barvu povrchu tělesa, viditelnou díky rozptýlenému světlu ve scéně.
- **Difúze (angl. diffuse).** Tato vlastnost udává barvu povrchu tělesa, která je viditelná díky přímému osvětlení.
- **Odlesky světel (angl. specular) a lesklost (angl. shininess).** Určuje jak moc budou vidět na povrchu odlesky od světel a na jakou barvu se přemění. Lesklost určuje jak ostrý (rozměrově malý) odlesk bude.
- **Odrazivost (angl. reflectance).** Určuje míru zrcadlení okolí od objektu.
- **Refrakce (angl. refraction).** Určuje míru průhlednosti materiálu.
- **Index Lomu (angl. refraction index).** Udává hustotu prostředí, kterým se šíří paprsek. Slouží především k výpočtům na rozhraní 2 prostředí, kdy se podle tohoto určuje pod jakým úhlem se paprsky budou lámat (u průhledných objektů).
- **Rozptyl odrazu (angl. diffuse reflection)**. Také matnost povrchu. Pouze u distribuovaného ray tracingu. Udává míru nepřesnosti odrazů paprsků. Čím matnější povrch, tím hůře lze rozeznat okolí, které se zrcadlí.

- **Rozptyl Refrakce (angl. diffuse refraction).** Udává míru nepřesnosti lomu paprsků na povrchu tělesa. Čím vyšší nepřesnost, tím hůře lze rozeznat okolí, které se v průhledném materiálu láme.

2.5.2 Phongův osvětlovací model

Tento model slouží jako náhrada fyzikálního modelu světla. Základní model počítá se 3 složkami, Uvedu ale rozšířený model, použitý v ray traceru, který pracuje i s odraženým a lomeným světlem.

Pro výpočet složek se používají vektory:

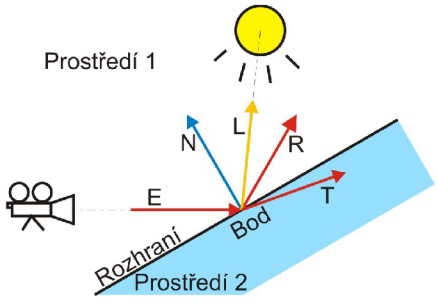
$\vec{E}$ - Jednotkový vektor pohledu (příchozího paprsku do počítaného místa).

$\vec{N}$ - Jednotkový vektor normály, kolmý na povrch tělesa.

$\vec{L}$ - Jednotkový vektor o směru od počítaného bodu k světlu.

$\vec{R}$ - Jednotkový vektor se směrem odraženého paprsku.

$\vec{T}$ - Jednotkový vektor se směrem lomeného paprsku.



Základní složky a rekurzivní složky Phongova modelu:

- **Ambient složka** určuje barvu vzniklou rozptýleným světlem v prostředí. Používá se k umělému doplnění světla ve scéně, čímž dělá lépe viditelnými objekty s malým koeficientem difúze a odlesku při špatném osvětlení. Tento nereálný parametr modelu nahrazuje z reality to, že každý osvětlený objekt se stává také zdrojem světla. Rozumné nastavení je přibližně 0,15 (15%) pro středně osvětlené scény.

$$I_a = i_a k_a$$

kde I_a je výsledná složka, i_a je koeficient celé scény a k_a konstanta **ambient** materiálu v podobě barvy.

- **Difúzní složka** určuje barvu vzniklou přímým působením světla na objekt. Je nezávislá na směru pohledu a počítá se pro každé světlo zvlášť. Barva každé části objektu je ale touto

složkou ovlivněna jinak a to v závislosti na úhlu mezi vektorem $\vec{N}$ a vektorem $\vec{L}$. Vzorec poté vypadá následovně:

$$I_d = I(\vec{L} \cdot \vec{N}) k_d$$

kde I_d je výsledná barva složky, I je barva světla a k_d je konstanta **difúze** materiálu v podobě barvy.

- **Zrcadlová složka** reprezentuje odlesky světel od povrchu objektu. Odlesky se vyskytují na povrchu lesklých hladkých ploch. Počítá se pro každé světlo zvlášť. Tato složka bude pro dané místo na objektu z jiných úhlu pohledu jiná.

$$I_s = I k_s \cos^{\alpha}(\varphi) = I(\vec{E} \cdot \vec{R})^{\alpha} k_s$$

kde I_s je výsledná barva složky, I je barva světla, k_s je konstanta **specular** materiálu v podobě barvy, α určuje ostrost odlesků a φ je úhel mezi $\vec{R}$ a $\vec{E}$.

- **Odražená resp. lomená složka** se vezme, tak jak je vrácena rekurzivním paprskem a jen se ovlivní koeficientem odrazivosti k_r resp. průhlednosti k_t. Výsledná složka bude mít označení I_r pro odrazy resp. I_t pro lomy.

Po vypočtení všech složek se tyto složky sečtou do výsledné barvy I_v.

$$I_v = I_a + I_r + I_t + \sum_{I\,\text{světla}} \left[I(\vec{E} \cdot \vec{R})^{\alpha} k_s + I(\vec{L} \cdot \vec{N}) k_d \right]$$

Ilustrace 4: Vzorec Phongova modelu osvětlení

2.5.3 Textury materiálu a jejich mapování

Materiály a výpočet jejich barev se počítá pro celé těleso, tedy každý objekt má přiřazenu jednu barvu. To lze změnit pokud přidáme do materiálu 2D texturu v podobě obrázku, nebo 3D procedurální texturu v podobě vzorce. V programu jsem použil pouze 2D textury pro plochy (roviny, obdélníky) a pro koule a tak se o nich zmíním.

2D texturu je potřeba vhodným obalit na 3D objekt. Způsobu jakým se toto vykonává, se říká mapování textur. Existuje větší množství druhů mapování, nejznámější jsou asi **mapování UV** a UVW. Pro projekt jsem vybral jednoduché mapování UV, které využívá vzorce, který vypočte ze 3D souřadnice 2D souřadnici v textuře.

V případě **plochy** je to jednoduché, stačí, když vypočteme pozici průsečíku od počátku plochy a podle toho určíme, který bod z textury namapujeme. Tomuto se říká **plošné mapování**.

V případě **koule** je způsobu vícero, jak namapovat texturu. Asi nejjednodušší a zároveň dobře použitelný způsob je **sférické mapování**, kdy textura oblepí celou kouli dokola. Ke kouli doporučuji podívat se na [5].

2.6 Výhody a nevýhody klasického ray tracingu

Klasický ray tracing je velmi elegantní, přímočará metoda vykreslování trojrozměrných scén.
Mezi její hlavní výhody patří:

- Velmi kvalitní výstup.
- Metoda je dobře implementovatelná a rozšiřitelná.
- Velmi přesné zobrazení, které je limitováno jen přesností výpočtu (př. architektura).
- Je ideální pro výstup 3D návrhů domů, aut, apod.
- Velmi perspektivní do budoucna (použití ve hrách apod.)
- funguje jako zjednodušený upravený abstraktní model reálného světla (na rozdíl od principu 3D grafiky zobrazované pomocí OpenGL)
- Dá se optimalizovat v určitých částech (např. z kvadratické náročnosti na lineární, příp. Urychlit renderování)

Hlavní nevýhody:

- Zatím se moc nehodí na interaktivní zobrazování (první pokusy s enginem Quake 3)
- S náročností scény roste doba výpočtu velmi rychle (dá se částečně řešit optimalizacemi).
- Pouze ostré stíny z bodových světel.
- Pouze dokonalé zrcadlové odrazy okolí.
- Pouze dokonale průhledné lomy okolí.
- Světlo neprochází přes průhledný objekt a tedy neozařuje objekty za ním.
- Světlo odražené zrcadlovým povrchem také neozařuje okolí.
- V porovnání ve výkonu s OpenGL při jednoduchých podobných scénách s podobnou kvalitou výstupu je extrémně pomalá (např. 1fps vs. 200fps)

Většinu nedostatků co se týče kvality se dá odstranit implementací dalších metod do ray traceru. Například distribuovaný ray tracing (měkké stíny, matné povrchy), radiosita (objekty se stávají samy zdroji světla), fotonové mapování (světlo svítí skrz průhledné objekty a odráží se od zrcadel), zpětné trasování paprsku (difrakce světla).

V následující kapitole se zaměřím na nadstavbu ray tracingu v podobě distribuovaného ray tracingu a jeho nových vlastností.

3 Distribuované sledování paprsku

Minulá kapitola probírala především základní (klasický) ray tracing, v jejímž závěru bylo uvedeno poměrně dost nevýhod. Obecně by se dalo říci, že díky přesnému matematickému zpracování scény se z ní ztrácí realističnost. Nejvíce klasické metodě ray tracingu schází nepřesnosti povrchů, nedokonalosti odrazů okolí a efekty způsobené omezenými vlastnostmi očí a kamer. Obraz pak vypadá nepřirozeně a dalo by se říci, jak z jiného světa. Právě zpracování těchto jevů lze do ray tracingu zakomponovat, pomocí náhodnosti v podobě distribuovaného ray tracingu.

Tato metoda na místo jednoho výpočtu paprsku spočítá paprsků více (náhodně rozložených) a všechny zprůměruje do výsledku. Tato distribuce probíhá u každého efektu na jiných místech Výsledek je potom rozmazaný, nepřesný a díky tomu vypadá přirozeněji.

Mezi efekty, které lze touto technikou simulovat patří:

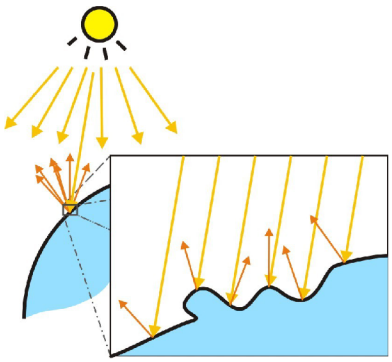
- Matné povrchy (rozptýlené odrazy, angl. diffuse reflection)
- Průsvitné materiály (rozptýlené refrakce, angl. diffuse refraction)
- Měkké stíny (angl. soft shadows)
- Hloubka ostrosti (angl.depth of field, zkratkou DOF)
- Rozmazání pohybem (dále jen angl. motion blur)

Program s těmito prvky je potom schopný generovat i obrázky, které člověk nerozezná od skutečných fotek. V závislosti na počtu vyslaných paprsků je potom také obraz kvalitní. Při malém počtu vzorků je v obrazu velmi vysoká míra šumu, při velkém počtu vzorků je pak obraz v některých detailech k nerozeznání od skutečnosti.

V následujících podkapitolách popíšu jednotlivé efekty a jejich funkčnost.

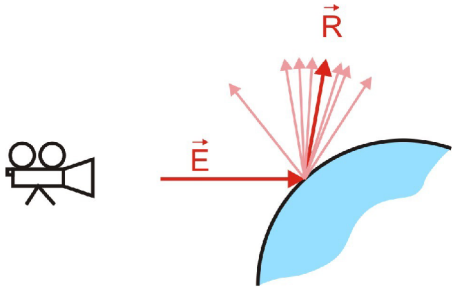
3.1 Matné povrchy

Klasický ray tracing umí zobrazovat jen přesné zrcadlové odrazy okolí a nereflektuje tak nepřesnosti na povrchu reálných objektu, které při pohledu vnímáme jako matný povrch.



Ilustrace 5: Odrazy na reálném povrchu

Distribuovaný ray tracing tuto vlastnost materiálu simuluje distribucí odražených paprsků, kdy místo jednoho odraženého paprsku jich vyšle více v podobném směru, jako dokonalý odraz, podle normálního rozložení s různým rozptylem. Tímto simuluje velmi mnoho náhodných vlivů, díky kterým je povrch nerovný. Rozptyl pak udává jak moc je povrch nerovný. V materiálu je reprezentován konstantou **diffuse reflection.**



Ilustrace 6: Distribuce odražených paprsků

Materiály pak působí měkkým až drsným povrchem v závislosti na nastavení scény. Pro vylepšení můžeme distribuované paprsky váženě průměrovat. Potom paprsky, které svírají menší úhel k referenčnímu paprsku odrazu mají vyšší váhu, než ty, které svírají s tímto paprskem úhel větší. Díky tomu nám pak stačí méně vyslaných paprsků na stejně kvalitní výstup.

3.2 Průsvitný materiál

Klasický ray tracingu zobrazuje jen dokonale ostré refrakce. Nelze tedy simulovat sklo s drsným povrchem ani tenkou vrstvu plastu. Což u některých scén ubírá na realističnosti.

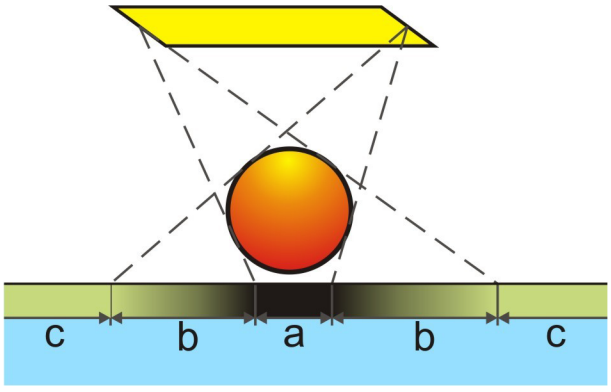
V distribuovaném ray tracingu lze simulovat nepřesnosti povrchu a tedy i průsvitnost materiálu. Lze tak velmi dobře napodobit např. povrch drsné skleněné desky. Lze také napodobit objekty z neprůhledných materiálů, které lehce prosvítají. Způsob simulace ale vůbec neodpovídá realitě, kdy se paprsky velmi mnohokrát lámou uvnitř materiálu. V distribuovaném ray tracingu totiž paprsky procházejí materiálem jen přímočaře a nelámou se. I tak to ale na napodobení průsvitných materiálu stačí.

Tato technika je totožná s technikou matných povrchů, jen se distribuují paprsky lomené, ne odražené.

3.3 Měkké stíny

U klasického sledování paprsku lze použít jen bodové světlo, protože vektor udávající směr stínového paprsku ke světlu se vždy vypočte jako rozdíl z polohy světla a průsečíku v prostoru (tedy nenáhodně). Stíny jsou pak tvrdé a ostré. V průsečíku na povrchu mohou nastat jen 2 situace. Buď je bod ve stínu, nebo je osvětlený.

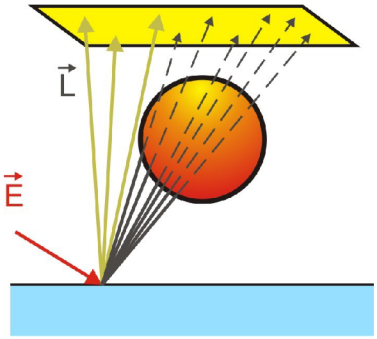
V reálném světě ovšem žádné nekonečně malé světlo neexistuje a můžeme se setkat jen se světly, které mají určitý tvar, rozměr a velikost. Nastávají pak situace, kdy z bodu na povrchu je viditelná jen např. polovina z celého světla a druhá půlka je zastíněná překážkou. Bod je pak napůl osvětlen a tomu odpovídá jeho světlost a barva. V závislosti na tom kolika procenty plochy světla je bod ozařován, je poté výsledná barva.



Ilustrace 7: Části s měkkými stíny

V zóně *a* je úplný stín, v zónách *c* úplně osvětlená plocha. V zónách *b* je měkký přechod ze stínu do osvětlené plochy. V této části jsou stíny měkké a přechází z úplně tmavého na žádný.

V distribuovaném ray tracingu se toto dá simulovat distribucí stínových paprsků. Místo jednoho stínového paprsku jich vrháme více. Tyto paprsky rovnoměrně distribuujeme do plochy světla, např. obdélníkového tvaru.



Ilustrace 8: Distribuované stínové paprsky

Některé paprsky dorazí ke světlu a jiné narazí na překážku. Pro každý paprsek se spočítá osvětlovací model a výsledek se zprůměruje ze všech těchto paprsků. Vzniknou pak různě tmavé stíny v závislosti na tom, jaké procento plochy světla daný bod ozařuje.

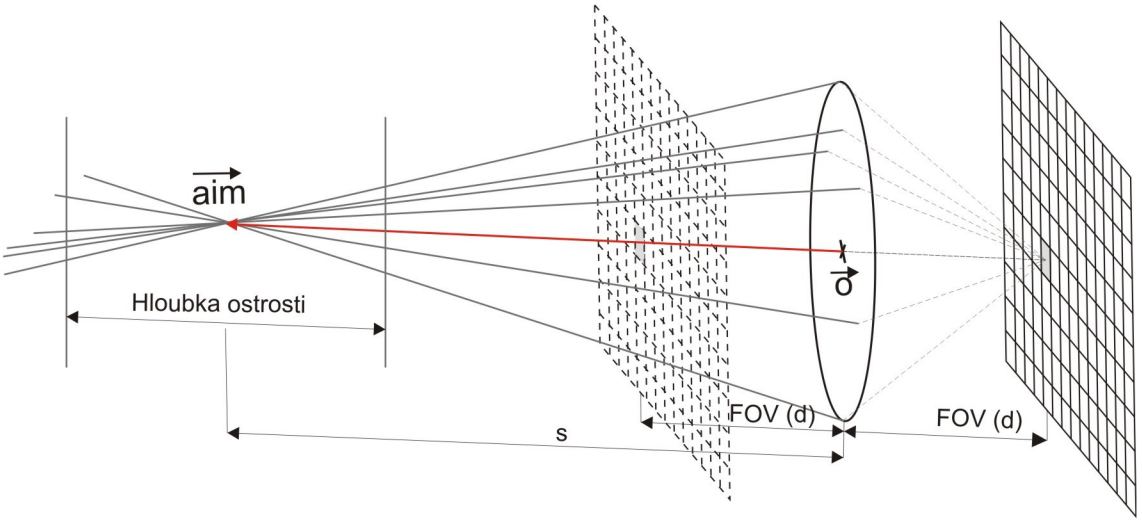
Tato technika má ještě další efekt a to ten, že se osvětlovací model pro průsečík počítá z více světelných míst nezávisle a pak se průměruje. Toto dodává i ploše bez stínu o něco reálnější vzhled, protože v reálném prostředí bod osvětluje více nezávislých paprsků a ty se o něj lámou až do projekční plochy.

3.4 Hloubka ostrosti

Hloubka ostrosti je velmi důležitý pojem ve fotografování. Vyjadřuje rozdíl vzdáleností nejbližšího a nejvzdálenějšího objektu, které jsou na výsledné fotografii přijatelně ostré. Projevuje se tak, že když „zaměříme“ oči na velmi blízký objekt, tak objekty více vzdálené se jeví jako rozmazané a naopak. Tento jev je dán tím, že velikost otvoru čočky oka, kterým prochází světelné paprsky dopadající na sítnici není nulová. Paprsky dopadají z prostředí na čočku a přes ní se lámou do projekční plochy, přičemž jen zaostřené věci „dopadnou“ na sítnici v podobě jednoho bodu. Vše co není zaostřené dopadá ostré před, nebo za sítnici a na sítnici se poté projevuje jako rozmazaný kruh o určitém poloměru.

V počítačové grafice je to právě naopak. Kamera je reprezentována pouhým bodem a „vidí“ vše naprosto ostře. Proto se efekt musí simulovat. V distribuovaném ray tracingu se dá simulovat

požitím zjednodušeného modelu čočky. V tomto modelu se nepočítá s fyzickými vlastnostmi kamer, ale pouze s kruhem rozptylu. I když je možné přepočítávat všechny hodnoty na reálné vlastnosti kamer, při renderingu zároveň např. s efektem motion blur by to způsobovalo zbytečně velké množství spočítaných informací navíc (pro každý pixel by se mnohokrát musely tyto parametry kamery přepočítávat) a výsledek by byl nakonec téměř stejný.



Generují se paprsky rovnoměrně v kruhu rozptylu. Pozice $\vec{o}$ je poloha kamery v prostoru a tečkovaná projekční plocha je ta přes kterou se vrhají paprsky. Pokud budeme používat techniku depth of field, tak místo z jednoho bodu vrháme paprsky z kruhu, rovnoběžného na projekční plochu, se středem v místě kamery. Určíme si poté referenční paprsek a přes jeho konec (určující zaostření) vrháme z kruhu náhodné paprsky. Výsledek poté zprůměrujeme.

Hloubka ostrosti nastává v určitých mezích od konce vektoru $\vec{aim}$ k projekční ploše i na druhou stranu. Toto rozmezí je dáno poloměrem kruhu rozptylu.

3.5 Motion Blur

Posledním efektem je motion blur (česky neobratně rozmazání pohybem). Jedná se o efekt, který způsobí při rychlém pohybu objektu, či kamery, rozmazání obrazu. Efekt vychází z reálných vlastností očí a kamer.

Ve fotoaparátech čip a závěrka, která určuje, kdy na čip bude dopadat světlo a kdy ne. V závislosti na tom jak dlouho je otevřená závěrka a jak je scéna osvětlena, dopadne na čip určité množství světla. Při slunečném počasí stačí pár milisekund, kdežto ve tmě je většinou potřeba sekund, k tomu aby byl obraz viditelně zaznamenám. Pokud se během doby otevření závěrky stane, že se nějaký objekt ve scéně pohne, tento pohyb se zaznamená do obrázku v podobě rozmazání. Velmi dobře jde tento efekt vidět při pozastavení filmu v rychlé scéně – obraz bude rozmazaný. Pokud se

ovšem díváme na sekvenci snímků, tak si tohoto rozmazání nevšimneme, protože mozek je na tento jev zvyklý (díky nedokonalým očím) a nevnímá ho.

V počítačové grafice nic takového samo o sobě není, protože obraz se generuje pro určitý diskrétní čas, ne pro interval. Pokud tedy tohoto v realitě většinou nežádoucího efektu chceme dosáhnout, musíme jej simulovat pomocí časově distribuovaného ray tracingu, kdy si nasimulujeme chování závěrky a snímáme scénu pro každý pixel vícekrát distribuovaně v náhodných časech podle simulované závěrky. Výsledek poté zprůměrujeme.

Pro toto je nutné mít nějakou pohyblivou scénu, případně pohybující se kameru.

3.6 Posloupnost výpočtů jednotlivých efektů

Všechny efekty ray tracingu je nutné určitým způsobem v ray traceru na sebe navrstvit. Nelze je z principu počítat vedle sebe nezávisle a je nutné počítat efekt z jiného efektu. Je to dáno tím jak se světlo šíří v reálu. Nejdříve dopadá světlo na objekty od nich se různě láme a odráží. Po čase některé paprsky světla dopadnou na povrch lidského oka (kamery). Tam se lomí přes čočku a dopadá na sítnici.

V distribuovaném ray traceru výpočet barvy bodu projekční plochy probíhá obráceně a to následovně:

- Začne počítat vzorky **motion bluru**, v každém z nich:
  - Zavolá výpočet vzorků **hloubky ostrosti**, v každém z nich:
    - Vypočte pro každé světlo **osvětlovací model** vzorků **distribuovaných stínových paprsků**
    - Zavolá výpočet vzorků **rekurzivních distribuovaných odrazů**, v každém z nich:
      - Vypočte pro každé světlo **osvětlovací model** vzorků **distribuovaných stínových paprsků**
      - Rekurzivně vypočte distribuované odrazy a lomy
    - Zavolá výpočet **rekurzivní distribuce lomů**, v každém z nich:
      - Vypočte pro každé **světlo osvětlovací model** vzorků **distribuovaných stínových paprsků**
      - Rekurzivně vypočte distribuované odrazy a lomy

Jak je vidět výpočty se do sebe hodně noří. Každý vnořený výpočet se počítá tolikrát, kolik je součin všech vzorků předchozích efektů nad ním. Pokud si nepřejeme počítat některé efekty, tak je

jednoduše z této posloupnosti vyjmeme a počítáme bez nich. Efekty vnořené do vyjmutého efektu se přesunou na úroveň vyjmutého efektu.

3.7 Výhody a nevýhody

Výhody distribuovaného ray tracingu:

- Velmi reálné efekty.
- Obraz je „měkčí“ a pro lidské oko hezčí.
- Velmi použitelné pro renderování filmů (rendering na clusterech).
- Založeno na zjednodušeném chování světla.
- Přímočará metoda, relativně ne složitý návrh (proti např. objemovým stínům)
- Všechny ostatní výhody klasického ray tracingu.
- Některé efekty lze velmi dobře optimalizovat.
- Velmi perspektivní do budoucnosti (optické procesory, chemické procesory).

Nevýhody:

- Řádově několikanásobná náročnost na čas výpočtu.
- Některé efekty lze jinou metodou (např. objemové stíny) se stejnými výsledky renderovat za zlomkový čas – ovšem za cenu velmi náročného návrhu a implementace.
- Chybí další reálné vlastnosti (např. světlo lámající se uvnitř sklenice, disperze na duhu)

4 Řešení distribuce paprsků

V této kapitole vysvětlím nejdříve nutné základy matematiky pro náhodné výpočty a poté vysvětlím generování náhodných paprsků.

4.1 Řešení náhodnosti

V klasickém ray tracingu se počítá pouze empiricky, tedy konečně přesně. Pokud spustíte rendering scény znovu, dostanete totožný výsledek. V distribuovaném ray tracingu se přidává náhodnost v podobě distribuce paprsků. Dvě totožné nezávisle vyrenderované scény se budou vždy lišit (zrněním, barvou) ve výstupním obrázku.

V následujících podkapitolách popíšu matematiku nutnou k náhodné distribuci.

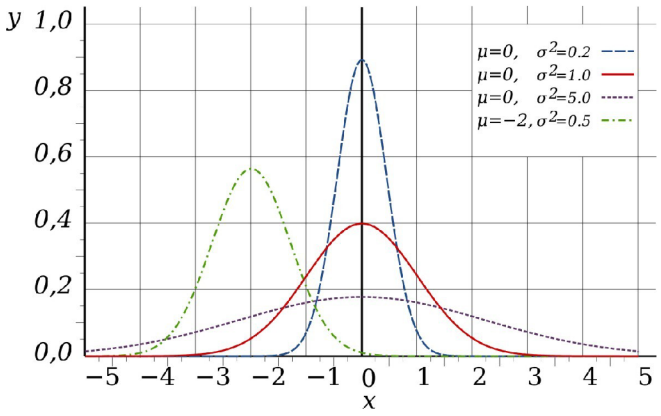
4.1.1 Rozložení pravděpodobnosti

Podle toho jaký náhodný jev chceme simulovat volíme odpovídající rozložení pravděpodobnosti.

Pro tento projekt se hodí následující:

- **Rovnoměrné rozložení** – U tohoto rozložení je pro všechny z možných jevů stejná pravděpodobnost, že nastanou. Jako příklad zmíním hrací kostku (6 stěn, 6 možných výsledků, všechny mají stejnou pravděpodobnost). V ray tracingu se dá využít k distribuci paprsků na čočce při simulování efektu hloubky ostrosti, nebo k distribuci paprsků v čase při simulování jevu motion blur.
- **Normální rozložení (nebo také Gaussovo)** – Je dáno Gaussovou křivkou, u které je obsah pod křivkou roven přesně jedné. Velmi se hodí pro náhodné jevy. Konkrétně pro simulování velmi složitých systémů jako je příroda, kdy není přesně zřejmé která náhodná veličina má vliv na daný stav (např. Proč je výška stromu větší o 0,5m než je průměr) a dá se tedy použít na distribuci paprsků. Například u nerovného povrchu totiž nelze přesně určit kudy a proč se o nerovnosti paprsek odrazí v povrchu až se dostane zpět ven, ale většina paprsků se odrazí podobným směrem.

Normální rozložení je dáno 2 parametry: střední hodnotou μ a rozptylem σ^2.Následující obrázek a vzorec ukazuje, jaký vliv mají parametry na tvar křivky hustoty pravděpodobnosti:



Ilustrace 9: Hustota normálního rozložení s různými parametry

$$f(x)=\frac{1}{\sigma\sqrt{(2\pi)}}\cdot e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

Ilustrace 10: Hustota normálního rozložení pravděpodobnosti

4.1.2 Distribuční a inverzní funkce

Distribuční funkce, pro každou reálnou hodnotu x, určuje jaká bude pravděpodobnost, že náhodná veličina X bude nabývat hodnoty menší nebo rovné x:

$$F(x)=P(X\leq x)$$

Je neklesající a nabývá hodnot z intervalu <0;1>. Distribuční funkce se vypočte z hustoty pravděpodobnosti následovně:

$$F(x)=\int_{-\infty}^{x} f(t)\,dt$$

Vzorec 4: Distribuční funkce

Inverzní funkce je taková, kdy vstupní parametr se stává výstupem a výstupní vstupem, např.:

$$y=f(x) \quad \rightarrow \quad x=f(y)$$

Generátor náhodných čísel s normálním rozložením pak lze zkonstruovat, tak že vytvoříme inverzní funkci k distribuční funkci normálního rozložení. Na vstup této inverzní funkce dáme výstup generátoru s rovnoměrným rozložením. Výstup z inverzní funkce pak bude generovat náhodná čísla s normálním rozložením.

4.2 Distribuce paprsků

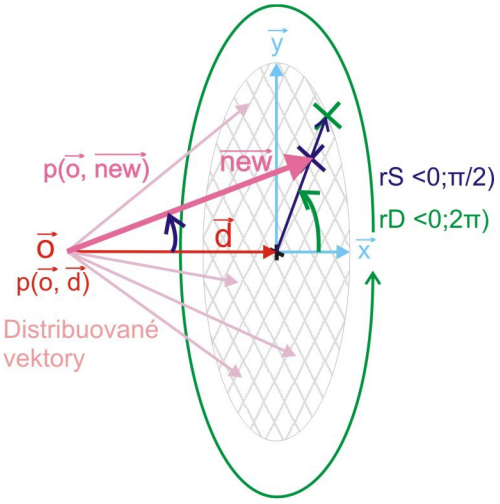
Následuje řešení distribuce paprsků pro jednotlivé efekty. Tyto řešení jsou naprosto stejným způsobem implementovány do programu.

4.2.1 Generování distribuovaných rekurzivních paprsků

Tento způsob generování se hodí pro rekurzivní efekty, tedy matné povrch, a průsvitné materiály. Paprsky se negenerují úplně náhodně, ale podle určitého referenčního paprsku, konkrétně odraženého či lomeného.

Toto se děje v určitých mezích. Poměrně dobře napodobuje reálné odrazy od matných materiálů náhodný rozptyl paprsků v pomyslné kapce (či kuželu) s normálním rozložení. Konkrétně pro směr odchýlení (tedy rotaci kolem referenčního paprsku) se volí rovnoměrné rozložení, které zajistí že se budou paprsky do všech směrů distribuovat stejně (rotační úhel). Pro odchylku (úhel odchylky) se použije normální rozložení, které zajistí, že paprsky, blížící se k referenčnímu budou vrhány s větší pravděpodobností. Parametr rozptylu normálního rozložení zajistí, že pro hodně matné materiály budou paprsky hodně rozptýlené do okolí a u hodně odrazivých materiálů se bude většina blížit původnímu paprsku a obraz pak bude ostrý, ale ne nereálně dokonale. Uvedu příklad výpočtu takto generovaného paprsku.

Mějme odražený paprsek p s pozicí $\vec{o}$ a směrovým vektorem $\vec{d}$ a budeme chtít kolem něj generovat další paprsky pomocí normálního rozložení s rozptylem σ=3; rozptyl určuje matnost materiálu; nakonec vypočteme výslednou barvu:



- Nejdříve je nutné zjistit lokální souřadné osy (vektory) $\vec{x}, \vec{y}$, které jsou kolmé na $\vec{d}$. To zajistíme díky pomocnému vektoru $\vec{tmp}$, který si zvolíme na jedné z hlavních

souřadných os, tak aby by svíral co největší úhel s $\vec{d}$:

$$\begin{aligned} &pokud\ min(\vec{d}_x, \vec{d}_y, \vec{d}_z) \equiv \vec{d}_x \quad \Rightarrow \quad \vec{tmp} = (1,0,0)\ jinak \\ &pokud\ min(\vec{d}_x, \vec{d}_y, \vec{d}_z) \equiv \vec{d}_y \quad \Rightarrow \quad \vec{tmp} = (0,1,0)\ jinak \\ &pokud\ min(\vec{d}_x, \vec{d}_y, \vec{d}_z) \equiv \vec{d}_z \quad \Rightarrow \quad \vec{tmp} = (0,0,1) \end{aligned}$$

$$\begin{aligned} &\vec{x} = \vec{d} \times \vec{tmp}\ ; \quad |x| = 1 \\ &\vec{y} = \vec{d} \times \vec{x}\ ; \quad |y| = 1 \end{aligned}$$

- Vygenerujeme 2 náhodné úhly:
  - Náhodný úhel rD <0; 2π) určující rotaci paprsku kolem $\vec{d}$ s rovnoměrným rozložením.
  - Náhodný úhel rS <0;π/2) s normálním rozložením se středem v 0 a rozptylem 3 určující odklon od $\vec{d}$.
- Výsledný směrový vektor nového paprsku spočítáme pomocí rovnice:

  $$\vec{new} = \vec{d} + (\cos(rD) \cdot \vec{x} + \sin(rD) \cdot \vec{y}) \cdot \tan(rS) \qquad |\vec{new}| = 1$$

  Kde nám tangens úhlu rS zajistí že se opravdu bude generovat s možným maximálním reálným odklonem 90°.
- Výsledný paprsek *n* bude mít potom tvar:

  $$n : \vec{o} + t \cdot \vec{new}$$
- Spočítáme osvětlovací model a jeho barvu.
- Barvu přičteme k celkové barvě bodu vynásobením koeficientem váhy $\cos(rS)$, což zajistí, že paprsky blížící se k referenčnímu budou mít ve výsledku větší váhu. Tuto váhu také přičteme k celkové váze.
- Celý tento postup opakujeme tolikrát, kolik vzorků tohoto efektu chceme provést.
- Výsledná barva se po přičtení všech distribuovaných barev podělí součtem všech váhových koeficientů paprsků.

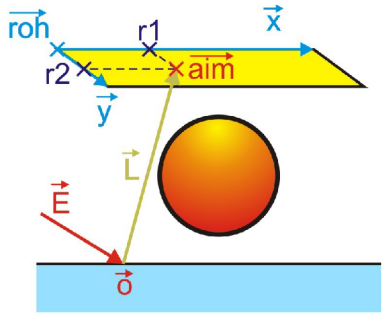
4.2.2 Generování distribuovaných stínových paprsků

Toto generování se využívá při výpočtech měkkých stínů popsaných výše.

Nyní potřebujeme generovat paprsky rovnoměrně směřující pouze do určité např. obdélníkové plochy (světla), protože jen tam se mohou uplatnit. Nejjednodušší a zároveň nejrychlejší způsob generování je ten, ve kterém matematicky popíšeme tvar světla a poté jen pomocí generátoru rovnoměrného rozložení na vstup tohoto popisu dáváme náhodná čísla a popis nám vrátí bod na světle. Například Obdélníkové plošné světlo se dá popsat pomocí pozice světla v podobě jednoho

rohu obdélníku ($\vec{roh}$) a dvěma vektory určující směr a velikost hran (lokální souřadné osy) ($\vec{x}$, $\vec{y}$).

Příklad výpočtu distribuovaného stínového paprsku a výsledné barvy pro obdélníkové světlo:



- Vygenerujeme 2 náhodná čísla: r_1 a r_2 v rozsahu <0; 1>.
- získáme vektory $\vec{aim}_x$ a $\vec{aim}_y$:

$$\vec{aim}_x = \vec{x} \cdot r1$$
$$\vec{aim}_y = \vec{y} \cdot r2$$

- Přičtením těchto vektorů k pozici světla $\vec{roh}$ získáme cílový bod distribuovaného stínového paprsku na světle $\vec{aim}$:

$$\vec{aim} = \vec{roh} + \vec{aim}_x + \vec{aim}_y$$

- Výsledný distribuovaný stínový paprsek n jednoduše vypočteme z bodu $\vec{aim}$ a $\vec{o}$:

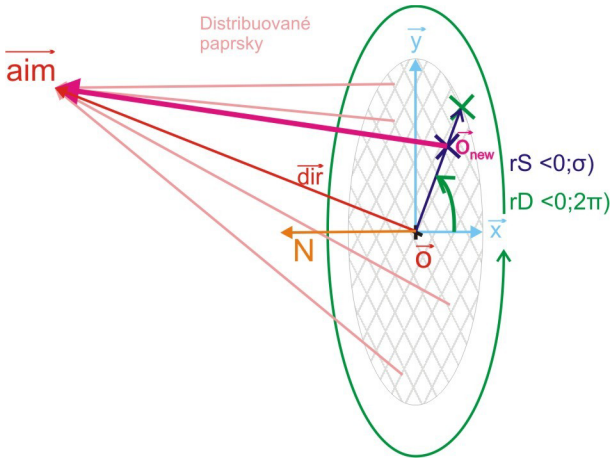
$$n : \vec{o} + t \cdot (\vec{aim} + \vec{o})$$

- Spočítáme osvětlovací model a jeho barvu pro tento paprsek.
- Barvu přičteme k výsledné barvě.
- Celý tento postup opakujeme tolikrát, kolik vzorků tohoto efektu chceme provést.
- Výslednou barvu nakonec podělíme celkovým počtem generovaných paprsků.

Kulové světlo je založeno na stejném principu. Bohužel se na povrchu kulového tělesa velmi špatně náhodně generují body a proto jsem toto zjednodušil na generování bodů v kruhu. Tento kruh se získá tak, že se zjistí paprsek od bodu do středu kulového světla, vytvoří se lokální osy $\vec{x}$, $\vec{y}$ a ty reprezentují poté reprezentují kruh, ve kterém se budou náhodné body generovat (viz generování distribuovaných rekurzivních paprsků). Délka těchto vektorů bude stejná jako je poloměr koule a tak bude pomyslný kruh přesně pokrývat kulové světlo. Zbytek výpočtů je stejný s obdélníkovým světlem.

4.2.3 Generování distribuovaných paprsků hloubky ostrosti

U tohoto efektu potřebujeme generovat počátky distribuovaných paprsků $\vec{o_{new}}$ v kruhu se středem $\vec{o}$. Tento kruh je rovnoběžný s projekční plochou a má střed v místě kamery. Cíl $\vec{aim}$ zůstává stejný pro všechny paprsky generované pro jeden bod projekční plochy. U dalších bodů projekční plochy se postupně mění směr referenčního paprsku $\vec{dir}$ a s ním $\vec{aim}$.



Postup generování a výpočet výsledku bude následovný:

- Zjistíme lokální osy (vektory) $\vec{x}$, $\vec{y}$ kruhu, tak aby byl kruh rovnoběžný s projekční plochou a měl střed v bodě kamery. K tomu nám pomůže vektor směru pohledu kamery $\vec{N}$, který ukazuje přesně do středu projekční plochy.

$$\begin{aligned} &pokud\ min(\vec{N}_x,\vec{N}_y,\vec{N}_z)\equiv\vec{N}_x \quad \Rightarrow \quad \vec{tmp}=(1,0,0)\ jinak \\ &pokud\ min(\vec{N}_x,\vec{N}_y,\vec{N}_z)\equiv\vec{N}_y \quad \Rightarrow \quad \vec{tmp}=(0,1,0)\ jinak \\ &pokud\ min(\vec{N}_x,\vec{N}_y,\vec{N}_z)\equiv\vec{N}_z \quad \Rightarrow \quad \vec{tmp}=(0,0,1) \end{aligned}$$

$$\begin{aligned} \vec{x}&=\vec{d}\times\vec{tmp}\,; \quad |x|=1 \\ \vec{y}&=\vec{d}\times\vec{x}\,; \quad |y|=1 \end{aligned}$$

- Vygenerujeme náhodný úhel a koeficient síly:
  - Náhodný úhel rD <0; 2π) určující rotaci paprsku kolem $\vec{dir}$ s rovnoměrným rozložením.
  - Náhodné číslo rS <0;σ) s rovnoměrným rozložením určující odklon od $\vec{o}$.
- Výsledný počátek nového paprsku spočítáme pomocí rovnice:

$$\vec{o_{new}}=\vec{o}+\left(\cos(rD)\cdot\vec{x}+\sin(rD)\cdot\vec{y}\right)\cdot rS$$

- Dále je potom nutné upravit směrový vektor, který se tím také změnil:

$$\vec{dir}_{new}=\vec{aim}-\vec{o_{new}}$$

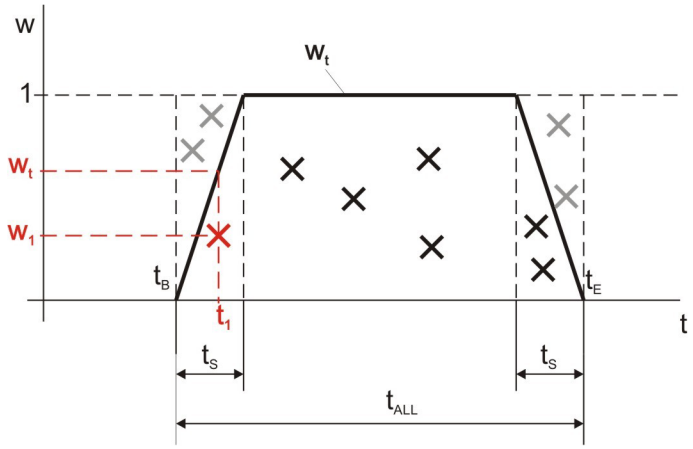
- Distribuovaný paprsek pak bude mít tvar:

$$n : \vec{o}_{new} + t \cdot \vec{dir}_{new}$$

- Spočítáme barvu pomocí osvětlovacího modelu pro daný paprsek.
- Barvu přičteme k výsledné barvě.
- Celý tento postup opakujeme tolikrát, kolik vzorků tohoto efektu chceme provést.
- Výslednou barvu podělíme počtem distribuovaných paprsků.

4.2.4 Generování časové distribuce pro motion blur

Při generování motion blur efektu nepotřebujeme generovat rozptýlené paprsky, ale časy ve kterých se bude scéna snímat. Pro toto si vytvoříme závěrku v podobě 4uhélníku, ve které tyto časy budeme generovat. Pravděpodobnost, že scéna bude vykreslena při otevírání či zavírání je menší než pravděpodobnost, že scéna bude vykreslena při plně otevřené závěrce. Při výpočtu se rovnoměrně náhodně zvolí bod v obdélníku ohraničeného t_B a t_E . Zjistí se jestli bod leží pod křivkou závěrky, pokud ano, tak se v daném bodě zjistí čas scény a váha barvy v tomto čase.



t je časová osa a w váhová osa určující podíl výsledku v daném čase na celkový výsledek. t_{ALL} je celkový čas otevření závěrky. t_S Je čas po který se závěrka otevírá, či zavírá a je dán koeficientem scény k_S <0;1>. k_s určuje poměr mezi otevíráním/zavíráním závěrky ku celkové době závěrky. t_B je čas počátku otevírání a t_E čas úplného zavření závěrky. t_1 pak udává náhodně zvolený čas, w_1 náhodně zvolené číslo a w_t váhu tohoto bodu ve výsledku (kopíruje závěrku).

Výpočet času scény pro efekt motion blur a výsledné barvy pixelu projekční plochy vypadá následovně:

- Vypočteme t_S z koeficientu k_S a t_{ALL} :

$$t_S = (t_E - t_B) * k_S / 2$$

- Vygenerujeme náhodné číslo t_1 v rozsahu < t_B ; t_E > určující pozici v čase.
- Pokud t_1 je v intervalu < t_B+t_S ; t_E-t_S > tak do tohoto času převedeme scénu a vyšleme paprsek a jeho barvu přičteme k celkové barvě bodu. Pokračujeme zcela novým výpočtem času.
- Vygenerujeme náhodné číslo w_1 v rozsahu <0,1>.
- Pokud t_1 je v intervalu < t_B ; t_B+t_S >, tak w_t vypočteme takto:

$$w_t = t_1 / t_S$$

- Pokud t_1 je v intervalu < t_E-t_S ; t_E >, tak w_t vypočteme takto:

$$w_t = 1-\left(t_1-(t_E-t_S)\right)/t_S$$

- Pokud je w_1 pod křivkou závěrky, tedy je-li menší než w_t , tak do tohoto času převedeme scénu a vyšleme paprsek a jeho barvu přičteme k celkové barvě bodu. Pokračujeme zcela novým výpočtem času.
- Pokud je ovšem w_1 nad křivkou závěrky, tedy je-li větší než w_t tak výpočet zahodíme a začneme novým.
- Na závěr podělíme celkovou barvu bodu celkovým počtem výpočtů/vzorků.

5 Návrh programu a jeho implementace

V této kapitole vás provedu návrhem programem a stručným popisem implementovaných prvků. Projekt byl vyvíjen v prostředích Windows i Linux (80% - 20%), program v IDE Eclipse, dokumentace byla psána v Open Office.

Program jsem implementoval celý v jazyce C++ pouze s použitím překladače GCC (G++), standardních knihoven, STL knihoven a fyzikální simulační knihovny ODE (Open dynamics engine). C++ jsem zvolil, protože jeho vlastnosti ho pro tento projekt předurčují:

- slušný výkon
- výborná přenositelnost
- slušná nižší náročnost na implementaci

C++ sice nedosahuje výkonu jazyku C a assembler, ale poskytuje výborné prostředky k implementaci (objekty, STL), které implementaci značně ulehčují, jsou stabilní a vysoce optimalizované.

Ray tracing lze velmi dobře popsat objektovým přístupem a tak jsem celý program (až na pár vyjímek) napsal objektově orientovaným přístupem.

Knihovnu ODE jsem zvolil pro její velmi benevolentní a otevřenou licenci a slušnou přenositelnost, díky dostupným zdrojovým kódům a v projektu se používá k fyzikální simulaci scény.

Ray tracer byl vyvíjen postupně. Nejdříve jsem implementoval základní, velmi omezený klasický ray tracer, a do něj sem postupně přidával jednotlivé efekty, a funkčnost. Následuje návrh programu a poté jeho implementovaná funkčnost.

Seznam všech implementovaných funkcí programu naleznete v příloze 1. Manuál.

5.1 Návrh programu

Před začátkem implementace bylo nutné si dobře promyslet jak program bude vypadat, co bude jeho vstupem, co výstupem a jaké možnosti bude poskytovat.

Aby byl relativně použitelný, určil jsem si následující kritéria, která splňuje:

- Program umí renderovat obrázky pomocí metody distribuovaného i klasického ray tracingu.
- Ukládá výsledky do souboru typu BMP.
- Umí renderovat různé objekty (koule, kvádr, plocha, trojúhelník).
- Pro znovupoužitelnost načítá scénu ze souboru .
- Je přenositelný.
- Je bez GUI, kvůli spouštění na systému bez GUI (výkonné servery apod).

- Má možnost vstupního parametru udávající číslo výstupního obrázku ze scény a parametr určující jen náhledový rendering.

Po zkoumání a navrhování vznikl diagram tříd, ten se ale při implementaci značně změnil. Tato finální podoba diagramu následuje.

5.2 Diagram tříd

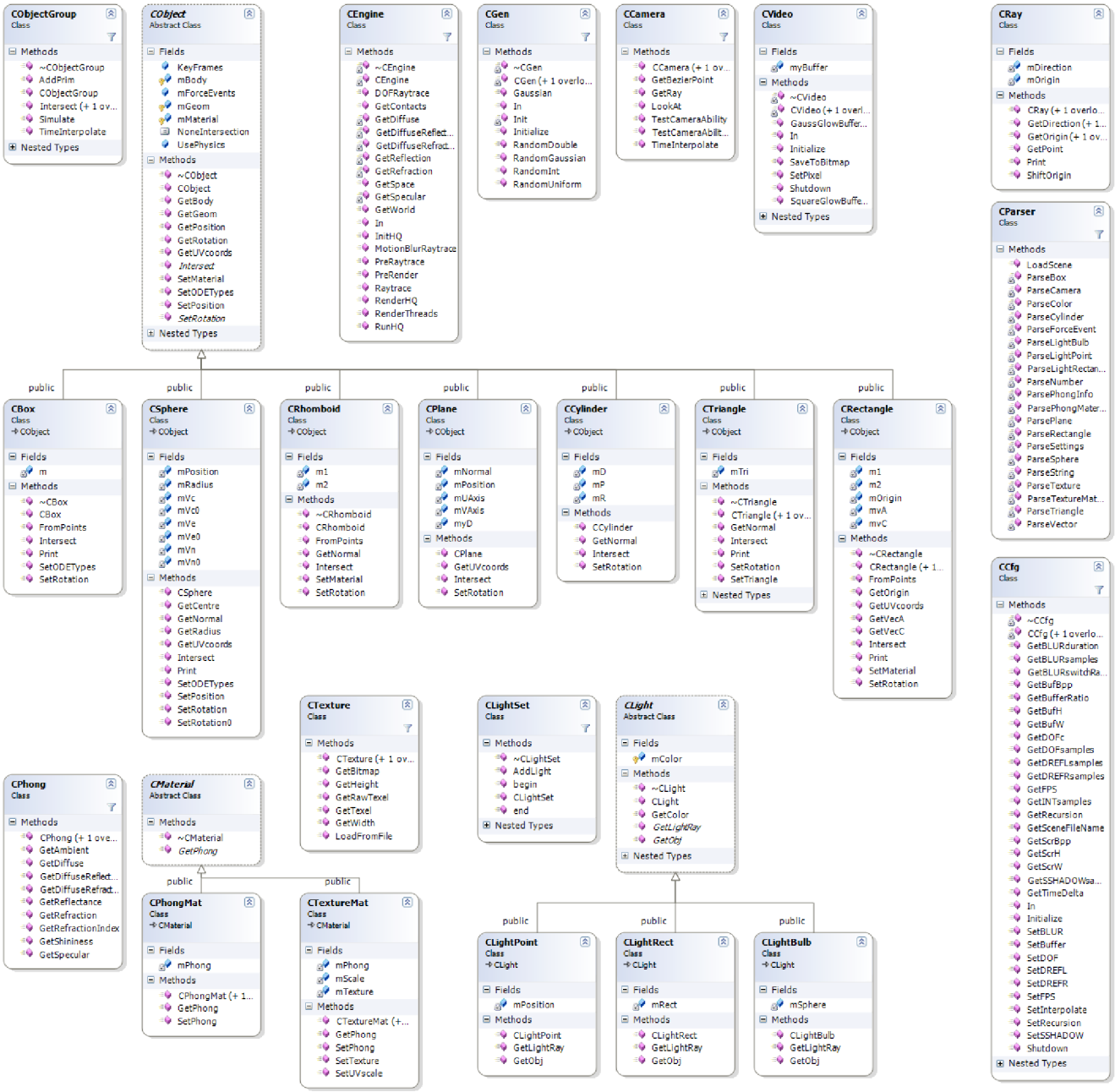


Diagram je velmi rozsáhlý a proto je zde uvedena jen jeho důležitější část. Úplný diagram je dostupný na přiloženém DVD

5.2.1 Singletony

Třídy CGen, CEngine, CVideo, CCfg, CLog jsou implementovány jako Meyersovy singetony. Jsou to třídy, které v celém programu mají jen jedinou instanci a více jich ani nemohou mít. Jsou dostupné odkudkoliv. Důvod proč jsem zvolil pro tyto třídy tento druh implementace je, že to velmi zjednoduší a zpřehlední kód při minimálních rizicích používání globálních proměnných.

5.2.2 Popis tříd programu

Program jsem se snažil psát přehledně a srozumitelně. Následuje tedy jen velmi stručný popis toho co jednotlivé třídy vykonávají. Pro podrobnější informace doporučuji shlédnout zdrojové kódy a Manuál v příloze, ve kterém je přehledný seznam všech implementovaných funkčností.

CParser

Tato třída slouží k načtení scény, kamer, světel a nastavení ze souboru. V jedné smyčce se prochází soubor, pokud blok odpovídá správnému rozložení, otestuje se, který prvek má být načítán, potom se zavolá odpovídající funkce a ta celý blok převede na informace a ty uloží buď do nového objektu a ten do pole stejného typu, nebo do připraveného globálního nastavení. Pro podrobnější informace co se načítá, doporučuji přečíst si přílohu č. 1 „Manuál“je tam popsáno jak načítání funguje (z uživatelského pohledu).

CRay

Implementace paprsku. Vysvětlení je v kapitole 2.2.

CGen

Třída CGen slouží jako generátor náhodných čísel s rozloženími normálním a rovnoměrným. Dále počítá Gaussovu křivku, což je potřebné pro interpolační matici. Všechny funkce generující náhodná čísla byly převzaty z [7].

CCfg

Slouží pro globální nastavení programu. Obsahuje informace o velikosti výstupního obrázku, veškeré globální nastavení týkající se scény (počty vzorků u jednotlivých efektů, délka trvání závěrky, síla rozmazání hloubky ostrosti apod.). Všechny tyto věci nastavuje do této třídy parser programu.

CVideo

Slouží k ukládání jednotlivých pixelů scény a ke ukládání výstupu do souboru. Spravuje pole barev, do kterého se ukládají jednotlivé pixely scény. Po skončení výpočtů pixelů, se pole barev převede na 24bitový BMP obrázek a ten se uloží do souboru s názvem odpovídajícího snímku ze scény.

CPhong

Tato třída shlukuje vlastnosti povrchu a materiálu. Obsahuje veškeré parametry potřebné pro světlovací model či pro distribuci paprsků. Všechny parametry jsou popsány a vysvětleny v kapitole 2.5.1.

CTexture

Implementuje do programu textury v podobě pole barev, každá textura je dána výškou a šířkou.

CMaterial

Je abstraktní třídou, ze které se odvíjí třídy CMaterialPhong a CMaterialTexture. První z nich obsahuje pouze objekt CPhong. Druhá obsahuje navíc texturu z které získává některé parametry a nahrazuje jimi při výpočtech některá nastavení uvedená v CPhong. Díky tomu je pak možné ve scéně zobrazovat na objektech barevné textury.

CObject

Reprezentuje těleso v prostoru, ke kterému je přiřazen určitý materiál, nastavení pozice pro různé časy a fyzikální vlastnosti. Od této třídy jsou odvozeny konkrétní objekty jako jsou např. koule a kvádr. Každý typ objektu má své vlastní algoritmy pro výpočet průsečíku s paprskem. Dále také některé další, např. získání UV souřadnic pro texturu apod. Každý z odvozených objektů má většinu funkcí přepsaných na jeho konkrétní tvar. Tyto objekty také obsahují funkce por získání své rotace a pozice v čase.

CHit

Tato třída je implementována jako podtřída CObject. Slouží pro předávání informací o průsečíku třídě CEngine, která následně tento průsečík zpracuje.

CObjectGroup

Tato třída slouží jako obálka všech objektů ve scéně, spravuje je, hledá nejbližší objekt pro zadaný paprsek, provádí simulaci scény pro určité rozmezí času a ukládá informace o poloze do objektů. Na popud nastaví všechny objekty scény do určité polohy a rotace na základě vstupního času.

CCamera

Velmi důležitá část ray traceru. Kamera udává komplexní nastavení kamery a tedy i projekční plochy v čase. Všechny kamery jsou uloženy ve zvláštním vektoru, které se předává do jednotlivých částí programu.

Kamera má uloženo více poloh a cílů a je dána počátečním a koncovým časem „videa“. Čísla snímků se z časů vypočtou vynásobením počtu snímků za s. V závislosti na tom ve kterém čase scény se renderuje, se pomocí beziérové křivky, která je dána těmito pozicemi resp. cíli, interpoluje nová pozice resp. nový cíl. Kamera se díky tomu plynule mezi jednotlivými snímky pohybuje a lze tak renderovat i jednodušší videa v podobě sekvence snímků. Obsahuje také počáteční a koncový čas v simulované scéně, který bude renderovat. Lze tak například v kameře renderovat pohyb věcí pozpátku, zpomaleně nebo zrychleně. Také obsahuje nastavení počátečního a koncového úhlu výhledu resp. bodu ostrosti. Tyto nastavení se poté pro jednotlivé snímky lineárně interpolují a lze tak plynule měnit v sekvenci například přiblížení, či zaostření. Dále také plynulé nastavení rotace kolem směru pohledu, které se také lineárně interpoluje.

CLight

Světla jsou dle diagramu rozlišeny do více podtříd (ClightPoint, CLightRect, CLightBulb) podle funkčnosti. Základní světlo funguje jednoduše tak ,že dostane na vstup pozici bodu v prostoru a vrátí paprsek obsahující tento bod a směr ke světlu. Daleko zajímavější jsou ovšem světla obdélníková a kulová, která samy počítají na svém povrchu cíle stínových paprsků, a jen vracejí enginu výsledný paprsek. Postup tohoto renderování byl popsán dříve.

CLightSet

Slouží jako obálka pro všechny světla ve scéně.

CEngine

Třída zpracovávající všechny výpočty týkající se ray tracingu a tvoří tak hlavní jádro programu.

Nejdříve se nastaví číslo snímku obrázku podle času a pustí výpočet obrázku pro daný čas. Začnou se procházet jednotlivé pixely projekční plochy a případně se ještě prochází každý pixel po subpixelech pomocí interpolační matice (Gaussovy) pro vyhlazený obraz.

U každého (sub)pixelu se zjistí relativní pozice a z té paprsek, který budeme vrhat do scény. Nastaví se pozice všech prvků ve scéně (kamery, objekty) a spustí se výpočet barvy pro daný pixel. V závislosti na nastavení efektů se zavolá buď funkce pro výpočet časové distribuce, nebo pro výpočet hloubky ostrosti, nebo metoda pro výpočet jediného paprsku. V každé volané funkci se dále spočítá osvětlovací model a případně zavolají rekurzivní funkce (odrazy, refrakce a jejich distribuované obdoby) v závislosti na hloubce rekurze. Podrobné vysvětlení generování je popsáno v kapitole 4.2 a podrobné volání jednotlivých výpočtů v kapitole 3.6.

5.2.3 Neaktivní kódy

Program dále obsahuje (či obsahuje ve starší verzi) dalších pár zajímavých funkcí, patří mezi ně: Glow efekt implementovaný pomocí distribuce přesvícených bodů do okolí (přes matici Gaussových koeficientů), vykreslování do okna, vícevláknové zpracování a také demonstrační třída zobrazující na výstup generované paprsky, a také Gaussovu křivku.

Tyto funkce, byly ovšem buď zakomentovány, jsou neaktivní, nebo byly z finální verze odstraněny, protože způsobovaly mírné potíže (zbytečné užití další knihovny, občasné artefakty při vícevláknovém zpracování) a jejich správná funkčnost nebyla prioritou.

6 Náročnost výpočtů

6.1 Klasický ray tracing

Tato metoda není relativně tak náročná, kvalitní výstupy můžeme dostat i do 30s, největším kámenem úrazu je ale rekurze.

Pokud by se rekurze vhodně neomezila, výpočet obrázku se scénou, ve které by byly všechny objekty odrazivé i průhledné, by byl neúnosně náročný a v extrémním případě by mohl trval i roky. Je to dáno tím, že počet paprsku se s hloubkou rekurze zvětšuje po geometrické řadě. Pro výpočet náročnosti jsem si vytvořil vzorec, který zohledňuje rekurzivní paprsky a také následně počítané stínové paprsky.

$$s_r = 2 \cdot w \cdot h \cdot l_c \cdot (2^r - 1)$$

s_r označuje kolikrát byla zavolána funkce pro výpočet paprsku, w šířka obrázku, h výška obrázku, l_c počet světel a r hloubka rekurze.

V tabulce uvádím 3 příklady jak by to mohlo vypadat (hodnoty jsou jen přibližné):

Rozlišení 800x600, 2 světla, vše je odrazivé a průhledné			
Hloubka rekurze	2	10	20
Počet paprsků	6 miliónů	2 miliardy	2 biliónů
Délka renderingu	30 s	166 m	116 dnů

Tabulka 1: Délka renderingu v závislosti na hloubce rekurze

Proto je vhodné nastavit rekurzi na přibližně 2-4, kdy jsou ve scéně zachovány detaily a náročnost není tak vysoká.

Je také možné optimalizovat náročnost tím, že zastavíme rekurzi dříve, pokud má rekurzivní paprsek na výslednou barvu jen malý vliv.

Další možností je zředěné vysílání paprsků, u jednolitých ploch a interpolace mezi nimi. Vytvoření stromu obálek pro objekty, kdy by se omezil počet spočítaných průsečíků.

Každá z těchto metod optimalizace bude různě úspěšná v závislosti na scéně a jejím obsahu.

6.2 Distribuovaný ray tracing

Náročnost v distribuovaném ray tracingu roste velmi rychle s počtem použitých efektů, komplexností scény a ještě více s velikostí rekurze. Vedle distribuovaného ray tracingu je v náročnosti klasický ray tracing s nadsázkou na úrovni „režijních nákladů“ toho distribuovaného. Vzorec by se dal pro celkový počet výpočtu průsečíku (včetně stínových) a výpočtů jejich osvětlovacích modelů pro 1 bod projekční plochy napsat takto:

$$S_R = mb \cdot dof \left(l_c sh + \sum_{n=1,\, r>0}^{n=r} 2 l_c sh (R+T)^n \right)$$
$$\left[\; ((R+T)>0 \;\vee\; r=0) \;\wedge\; mb, dof, l_c, sh > 0 \; \right]$$

S tím že se u každého bodu ve scéně počítá se všemi efekty. Při nastavení vzorků na hodnotu 1 lze výpočet považovat za výpočet klasického ray tracingu.

Pro celý obraz je pak počet paprsků:

$$s_r = S_R \cdot w \cdot h$$

Kde S_R je celkový počet paprsků pro jeden pixel ve scéně, mb počet vzorků motion bluru, dof počet vzorků hloubky ostrosti, sh počet stínových paprsků pro jeden osvětlovacího modelu, l_c počet světel ve scéně , r počet rekurzí, R počet distribuovaných odrazů , T počet distribuovaných lomů, w šířka obrazu v pixelech a h výška obrazu v pixelech.

Při výpočtech lze ale také počítat s tím, že počet vzorků dof násobí počty vzorků sh , R a T hodnotou. Díky tomu stačí nastavit např. 200 vzorků dof a 2 vzorky měkkých stínů a měkké stíny poté budou (v hlavně zaostřených oblastech) vypadat, jako by vzorků bylo přibližně 400. Toho lze využít a čas renderingu výrazně zkrátit při zachování slušné kvality.

Opět uvedu v tabulce pár příkladů jak by to mohlo vypadat (hodnoty jsou jen přibližné):

800x600, l_c = 2 (světla), vše je odrazivé a průhledné					
Hloubka rekurze (r)	1	2	1	2	2
Vzorků motion blur (mb)	32	32	100	100	100
Vzorků hloubky ostrosti (dof)	18	18	100	100	100
Vzorků měkkých stínů (sh)	2	2	2	2	8
Vzorků odrazů (R)	2	2	1	2	4
Vzorků lomů (T)	0	0	0	0	4
Počet paprsků	$5{,}5 \cdot 10^9$	$15 \cdot 10^9$	$57 \cdot 10^9$	$249 \cdot 10^9$	$11 \cdot 10^{12}$
Délka renderingu	2h	5,5h	21h	3,7 dnů	166 dnů

Tabulka 2: Náročnost distribuovaného ray tracingu v závislosti na nastavení efektů a rekurze

Z tabulky vyplývá, že tato metoda je extrémně náročná na výkon. Nejvíce se na rychlosti projeví množství odrazivých a průhledných ploch společně s rekurzí. Proto je vhodné využívat odrazivé a průsvitné/průhledné materiály jen minimálně, ne li vůbec a nastavit rekurzi maximálně na hodnotu 2-3.

Ve vlastním programu se tato extrémní náročnost tolik neprojevila, protože snižuji počty vzorků s rekurzí a scéna má většinou jen část obrazu s těmito náročným na výpočet plochami.

Časy byly přepočteny pomocí poměru počtu paprsků ze základního vzorku s časem 2 hodiny. Výsledky jsou tedy velmi nepřesné a mohou mít chybu i +- 50%. Ovšem pro představu složitosti to stačí. Program nepoužívá žádné optimalizace, s nimi by výpočty mohly být i o několik řádů rychlejší.

Výpočet probíhal na počítači s procesorem Intel C2D T8100 2,1GHz 3MB L2 cache.

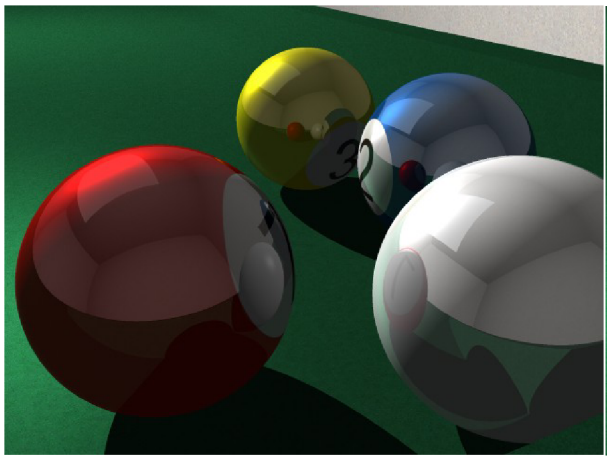
7 Dosažené výsledky

Hlavní cíl implementovaného programu je vytvářet obrazové výstupy. V následujících podkapitolách se pokusím tyto výsledky zhodnotit, případně porovnat distribuovanou techniku s klasickou a popsat výsledky mé práce. Všechny obrázky zde uvedené jsou v rozlišení 640x480. U všech obrázků klasického ray tracingu je použita 3x3 interpolace paprsků uvnitř jednoho pixelu.

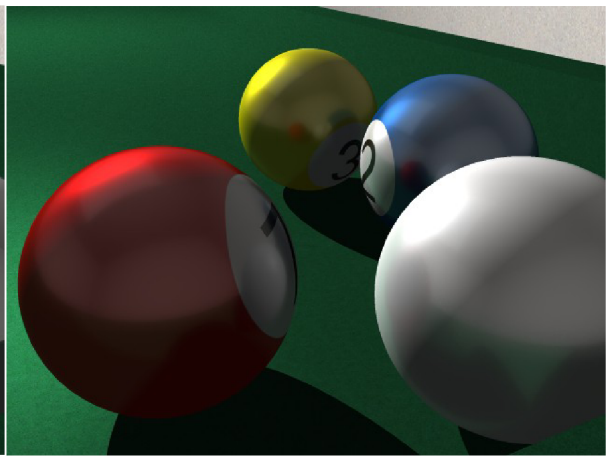
7.1 Matné povrchy

Tato technika distribuovaného ray tracingu, je velmi efektní a dodává předmětům nádech realističnosti díky na pohled změkčenému materiálu. Využívá distribuce zrcadlových odrazů a díky tomu je velmi náročná, protože její počet paprsků roste s rekurzí geometrickou řadou. Do programu se mi podařilo implementovat omezení, kdy s každou rekurzí počet vzorků klesá a díky tomu již není scéna tak náročná.

Následují obrázky pro porovnání klasické metody a té distribuované:



Ilustrace 11: Klasický ray tracing (1 rekurze, 1 minuta)

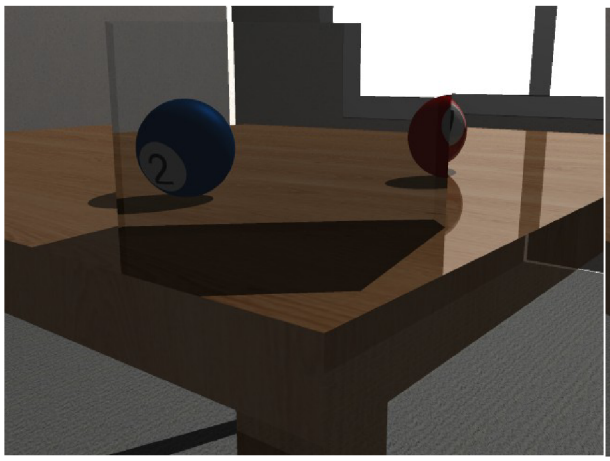


Ilustrace 12: Distribuovaný ray tracing: Matné odrazy (100 vzorků, 3x3 interpolace, 1 rekurze, 32 minut)

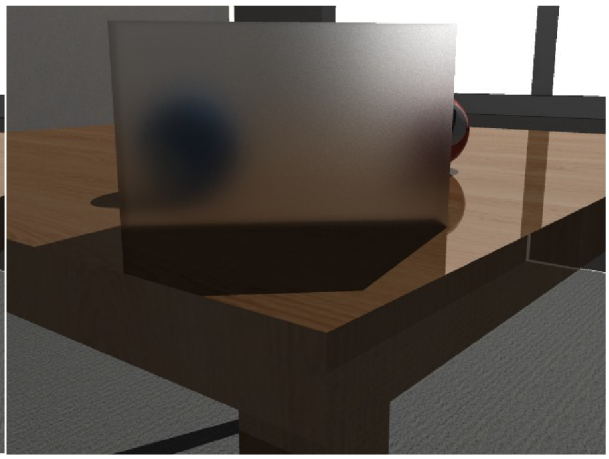
Z obrázků je vidět, že matné povrchy vypadají opravdu lépe a koule tak vypadají mnohem reálněji. Tento efekt dodává objektům živost. Bohužel náročnost s rekurzí a počtem objektů s odrazivým materiálem roste opravdu extrémně rychle. Ovšem při použití jen na jeden, či 2 předměty ve scéně s jedinou rekurzí dokáže výstup velmi obohatit scénu za cenu ne vysokých nároků. Nutno podotknout, že bez efektu měkkých stínů, je scéna velmi nepřirozená (matné povrchy, ale dokonale ostré stíny moc kontrastují) a výsledek tak může připadat na oko hůře než klasická metoda.

7.2 Průsvitné materiály

Tento efekt funguje na stejném principu jako minulý, ale s rozdílným výsledkem. Dokáže simulovat matné sklo, či průsvitný plast. V klasickém ray tracingu toto možné není. Výsledek vytváří dojem reálného matného skla. Náročnost tohoto efektu je ovlivněna rekurzí opravdu velmi hodně, protože je nutná o hodnotě minimálně 2. Na první pohled 40 vzorků na bod se zdá být málo, ale při rekurzi 2 to dělá na bod přibližně 1700 vzorků! A to už je opravdu hodně a výpočet je tak 40krát náročnější a pomalejší, než u jednoduché rekurze odrazu. Podařilo se mi ale implementovat omezovač, který téměř geometricky počet vzorků s rekurzí snižuje.



Ilustrace 13: Dokonalé průhledný materiál (klasická metoda: 2 rekurze 3 minuty)



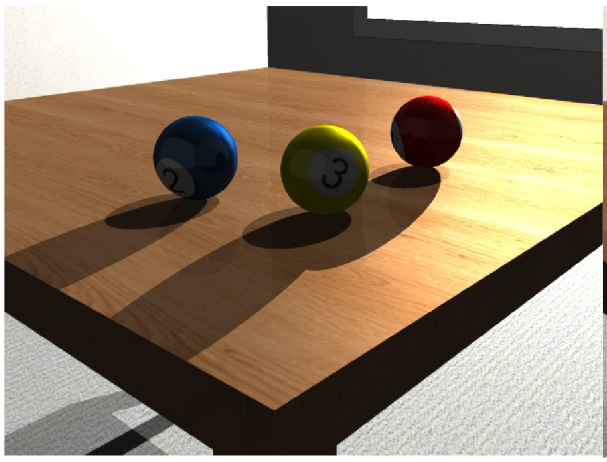
Ilustrace 14: Průsvitný materiál (distribuovaná metoda: 40 vzorků, 2 rekurze, 3x3 interpolace, 6 hodin)

Průsvitné objekty ve scéně vypadají velmi realisticky a proti nereálně průhlednému materiálu na levém obrázku působí velmi dobrým dojmem. Ovšem daní za tento efekt je opravdu velmi pomalý výpočet. Obrázek bez tohoto efektu se renderoval 1 minutu, kdežto ten samý s efektem (40vzorků) přibližně 6 hodin! A to ještě efekt nezabíral 100% plochy výstupu a v každé rekurzi se uměle počet vzorků snižoval. Tento efekt nemá takové využití jako předchozí a tak mu lze tuto náročnost prominout. Jeho využitelnost považuji za nejmenší ze všech implementovaných efektů i když v této scéně vypadá velmi dobře.

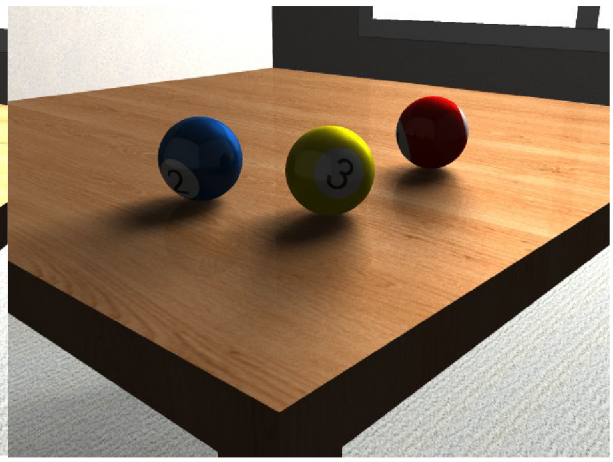
7.3 Měkké stíny

Efekt simuluje měkké stíny způsobené plošnými světly, kdy bod osvětluje jen část světla, a vnáší tak do scény vysokou míru realističnosti. Tento efekt nepůsobí jen na oblasti napůl osvětlené, ale také na ostatní oblasti, ve kterých jistou mírou přidává náhodnost do výpočtů Phongova osvětlovacího

modelu. Díky obojímu pak scéna vypadá měkce a přirozeně, na rozdíl od ostrých stínů, které renderuje klasická metoda.



Ilustrace 15: Klasický ray tracing (1 minuta)



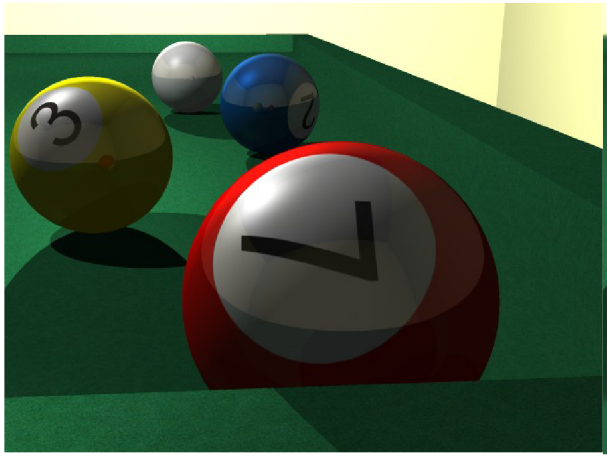
Ilustrace 16: Distribuovaný ray tracing: měkké stíny (32 vzorků, 2 světla, 3x3 interpolace 30 minut)

Tyto Ostré stíny klasické metody renderingu jsou do očí bijící a vyloženě obraz odsuzují k nereálnému zobrazení. Efekt měkkých stínů tento nedostatek plně potlačí a dokonce dodá do scény vyšší živost v podobě přirozenějších barev na všech površích, způsobené více výpočty osvětlovacího modelu pro dané body.

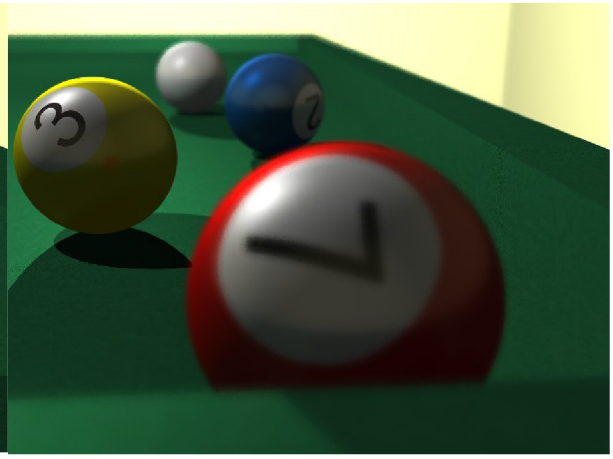
Měkké stíny považuji společně s matnými povrchy za velmi použitelné a potřebné efekty, bez kterých by obraz vypadal příliš uměle.

7.4 Hloubka ostrosti

Tento efekt simuluje reálné vlastnosti očí a výslednému obrazu dodává výborné hlavní i vedlejší vlastnosti. Jako hlavní efekt je hloubka ostrosti, kdy se do popředí dostávají jen důležité objekty. Tím vedlejším je excelentní antialiasing hran ve scéně. Považuji ho za ještě lepší než ten který jsem implementoval (interpolací gaussova 2D matice). Další výbornou vlastností pro rendering, kterou v něm vidím, je fakticky nižší náročnost výpočtů komplexní scény se všemi efekty pomocí velkého množství DOF vzorků a velmi malého množství ostatních (matné povrchy, refrakce). Náročnost roste totiž u DOF efektu jen lineárně, kdežto při zvyšování vzorků rekurzivních efektů roste náročnost geometricky. DOF totiž svým velkým počtem vzorků zajistí že první rekurze efektů bude mít velký počet vzorků, kdežto další, jen minimum, a scéna není pak tak náročná při zachování vysoké kvality.



Ilustrace 17: Klasický ray tracing (2 minuty)

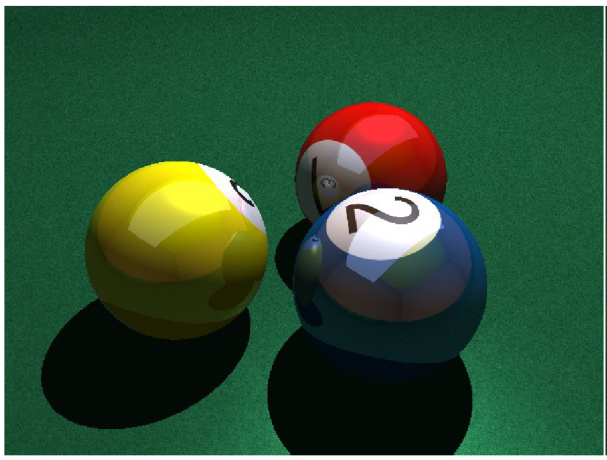


Ilustrace 18: Distribuovaný ray tracing: Hloubka ostrosti (200 vzorků, 0.35c) + difúzní odrazy (4 vzorky) (42 minut) (bez interpolace)

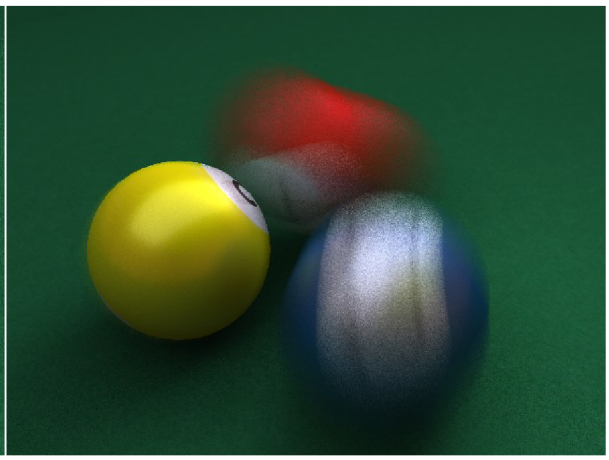
Na obrázcích je krásně vidět rozdíl při použití DOF efektu proti klasické metodě. Scéna nabývá daleko reálnějšího vzhledu, hloubky ostrosti a v kombinaci s matnými odrazy (jako na pravém obrázku) dává excelentní výsledek. pokud bychom použili navíc měkké stíny, scéna by možná vypadala téměř jako fotka. Navíc co se týče náročnosti, tak ta díky linearitě náročnosti DOFu opravdu není velká. Například proti průsvitným materiálům je velmi nízká při daleko více vzorcích a to i s dalšími.efekty.

7.5 Motion Blur

Tento efekt simuluje vlastnosti sítnice očí, či snímače kamery. Pro porovnání s klasickou metodou ray tracingu je potřeba se podívat na sekvenci snímku o vysokém fps, kdy již efekt nebude tak zřetelný, ale bude dodávat reálnější pocit ze vzhledu výstupu. Ovšem tento druh porovnání není moc dobře možný, když si uvědomíme, že 1s má normálně 25 snímků a jeden snímek se renderuje třeba 30 minut (2s = 1den) a chceme například video o délce alespoň 10s. Následuje tedy jen porovnání 2 obrázků:



Ilustrace 19: Klasický ray tracing (1 minuta)



Ilustrace 20: Distribuovaný ray tracing: Motion Blur (2fps, čas 2.5s, závěrka 0.5s, 80% závěrky plně otevřeno, 32 vzorků) + DOF (18 vzorků) + měkké stíny (2 vzorky) + matné povrchy (2 vzorky) (90 minut)

Při pohybujících se předmětech efekt dodává velmi reálné podání, hlavně v pomalejších sekvencích o dlouhé závěrce je efekt velmi zřetelný a s trochou nadsázky lze říci, že obraz vypadá stejně jako v reálu. Na obrázku jsou přidány všechny ostatní efekty (difúzní refrakce). Scéna vypadá proti standardnímu ray tracingu opravdu velmi dobře a velmi realisticky. V příloze 2 se nahází sekvence 16ti snímků této scény.

7.6 Kombinované efekty

Efekty distribuovaného ray tracingu při samostatném použití vypadají velmi nuceně a nepřirozeně. Je dobrý nápad efekty kombinovat a najít rovnováhu mezi velmi dobrým výstupem, použitými efekty a náročností výpočtu.

Jak je vidět obrázky vypadají opravdu velmi rozdílně, kdy distribuované sledování přináší realistické vlastnosti obrazu (alespoň proti výstupu bez efektů). Při renderování scén jsem zjistil, že největší efekt na scénu má kombinace měkkých stínů a matných povrchů. Dále je velmi vhodné používat DOF efekt, pro jeho vlastnosti uvedené výše (může snížit náročnost výpočtů při podobné kvalitě).

8 Závěr

Distribuované sledování paprsku je velmi dobrá metoda pro napodobování fotorealistických obrázků. V tomto projektu jsem tuto metodu podrobně nastudoval a popsal Poté jsem navrhl a implementoval program implementující prvky jak distribuovaného, tak klasického sledování paprsku (prolínají se). Obrazové výstupy programu jsou velice efektní a na pohled působí velice dobrým dojmem. Program, který jsem implementoval obsahuje poměrně velké množství implementovaných prvků, a na renderování jednoduchých scén je již relativně použitelný a to především díky načítání scény ze souboru a implementaci více druhů objektů a textur.

Rozsah tohoto projektu byl nejvyšší, na kterém jsem zatím pracoval a díky němu jsem získal mnoho informací a zkušeností z oboru programování i počítačové grafiky a byl tedy pro mě velmi přínosný.

Program bohužel není téměř vůbec optimalizován, a neobsahuje pokročilejší tvary (mesh, kvadriky, CSG), což může být dobrá myšlenka k rozvíjení současné práce, případně k pokračování současné práce v jiné, daleko složitější.

Tento semestr jeden student pracoval také na bakalářské práci věnované ray tracingu. Jedná se o síťově distribuovaný. Bylo by zajímavé se pokusit tyto 2 metody shodné názvem, ale odlišné funkčností propojit a využívat tak výhod obou, tedy kvalitních výstupů a rychlého renderingu na více počítačích současně.

Distribuovaný ray tracing mě velice za ujmul a přitáhl mě tak ke světu počítačové grafiky díky svým viditelným výsledkům v podobě kvalitních obrazů.

Literatura

[1] Appel, A. *Some Techniques for Shading Machine Renderings of Solids. SJCC,* 1968

[2] Cook, R. *Distributed ray tracing.* ACM, 1984
Dokument dostupný na URL
http://portal.acm.org/citation.cfm?id=808590

[3] Owen, G., S. *HyperGraph: Computing a Reflected Ray,* 1998
Dokument dostupný na URL
http://www.siggraph.org/education/materials/HyperGraph/raytrace/rtreflec.htm

[3] Greve, de, B. *Reflections and Refractions in Ray Tracing,* 2004
Dokument dostupný v podobě PDF na URL
http://www.flipcode.com/archives/reflection_transmission.pdf

[4] Haines, E., Akenine-Möller, T., Foscari, P. *3D Object Intersection,* 2007
Dokument dostupný na URL
http://www.realtimerendering.com/int/#I304

[5] Rademacher, P. *Ray Tracing: Graphics for the Masses*
Dokument dostupný na URL
http://www.cs.unc.edu/~rademach/xroads-RT/RTarticle.html

[6] Autor neuveden, *Normal distribution*, Wikipedia, 2008
Dokument je dostupný na URL
http://en.wikipedia.org/wiki/Normal_distribution

[7] Bourke, P. *Uniform random number generator*, 1998
Dokument dostupný na URL
http://local.wasp.uwa.edu.au/~pbourke/other/random/

[8] Shklyar, D. *To Photorealism and Beyond*, 2003
Dokument dostupný na URL
http://features.cgsociety.org/story_custom.php?story_id=1724&page=1

[9] Bikker, J. *Raytracing Topics & Techniques*, 2004
Dokument dostupný na URL
http://www.flipcode.com/archives/Raytracing_Topics_Techniques-Part_1_Introduction.shtml

[10] Phong shading, Wikipedia, 2008
Dokument dostupný na URL
http://en.wikipedia.org/wiki/Phong_shading

Příloha 1. Manuál

Co je ray tracer?

Ray tracer je program sloužící ke renderování velmi realistických, či nereálných obrazů na základě matematického popisu scény. Následuje popis použití a seznam implementovaných vlastností.

Použití

Program se spouští z příkazové řádky. Umožňuje předat parametry:

- `-p  --preview`:
  Parametr sloužící k dočasnému vypnutí všech efektů, což je velmi vhodné pro vykreslování náhledů v relativně krátkém čase.
- `-s  --scene <název souboru>`:
  Parametr sloužící k určení souboru scény.
- **-f --frame <číslo snímku>**:
  Parametr sloužící k nastavení, který snímek z časů kamery sem má vyrenderovat.

Příklady použití:

```
raytracer --preview
raytracer --frame 2
raytracer --scene scene.txt
raytracer -f 15 -s scéna.hhh
raytracer -p --frame 45 --scene sc1.src
```

Nastavení scény

Platí následující pravidla

☑ Veškerý popis se odděluje mezerami, případně mezerou, složenou závorkou a mezerou. Na počtu mezer nezáleží.

☑ Číslo se zadává s desetinnou tečkou (ne čárkou).

☑ Celá scéna je popsána prvky.

✗ Textové názvy nesmí obsahovat mezery a složené závorky.

Prvky jsou určené typem, jménem a atributy. Atributy jsou znovu prvky. V raytraceru se používají 3 druhy popisu prvku:

- S typem, jménem a atributy ve složených závorkách
  <*typ_prvku***>** **<**jméno_prvku**>** { **<**atributy_prvku**>**... }
- Základní prvek se jménem a atributy ve složených závorkách
  <jméno_prvku> { <atributy prvku>... }
- Základní prvek se jménem a atributem
  <jméno_prvku> <atribut>

Základní typy prvků použité jako atributy

Následují 3 nejzákladnější prvky, které jsou v nastavení použity.

- Název souboru *file* **filename** <název_souboru>
- Reálné číslo *real* <**název_atributu**> <číslo>
- Vektor *vector* <**název_atributu**> { <číslo> <číslo> <číslo> }

Pro přehlednost, budu používat dále jen jejich symbolické označení např.:

real **cislo**

což se převede při zápisu do souboru scény na:

cislo <reálné_číslo>

Nebo např. *vector* **position**

na **position** { <reál_číslo> <reál_číslo> <reál_číslo> }

Nebo např. *file*

na **filename** <název_souboru>

Pokročilé typy prvků a jejich použití v dalších prvcích

Mějme např. vlastnost materiálu:

phong_info **sklo** { různé atributy tohoto prvku .... }

Pokud ji budeme chtít použít jako atribut např. v prvku typu materiál, nebudeme opisovat celou její definici, ale jen uvedeme její typ a název, např.:

material_phong **skleněný_materiál** { *phong_info* sklo }

čímž docílíme, že materiál skleněný_materiál bude mít vlastnosti materiálu popsané ve sklo (phong_info).

Dále budu uvádět tyto již definované prvky použité jako atributy, jen jako <druh_prvku> <název_definovaného prvku>.

př.: *phong_info* <**název**>

Pokročilé typy prvků a jejich atributy použité k popisu materiálu

phong_info

phong_info <**název**> { atributy }

Udává vlastnosti materiálu. Má následující atributy:

- *vector* **ambient** – Barva neosvětleného objektu (projektu se násobí koeficientem 0.12).
- *vector* **diffuse** – Barva osvětleného objektu.
- *vector* **specular** – Barva odlesků světla.
- *real* **shininess** – Ostrost odlesků světla.
- *real* **reflectance** – Zrcadlovost materiálu.
- *real* **diffuse_reflection** – Rozptyl odrazů od povrchu.
- *real* **refraction** – Průhlednost materiálu.
- *real* **diffuse_refraction** – Rozptyl refrakcí.
- *real* ***refraction_index*** – Index lomu.

texture

texture **<název>** { atributy }

Určuje texturu. Atributy jsou:

- *file* – Název souboru s texturou.

material_phong

material_phong **<název>** { atributy }

Popisuje základní jednobarevný materiál. Atributy jsou následující:

- *phong_info* **<název>**

material_texture

material_texture **<název>** { atributy }

Popisuje materiál s texturou. Atributy jsou následující:

- *phong_info* **<název>**
- *texture* **<název>**
- *real* **scale_U** – Na kolika jednotkách ve scéně bude natáhlá textura do šířky.
- *real* **scale_V** - Na kolika jednotkách ve scéně bude natáhlá textura na výšku.

Pokročilé prvky použité k popisu objektů

add_force_event

Je to jediná vyjímka ze složitějších prvků, která je uvedena druhem, ale ne názvem a vypisuje se vždy až v objektu.

add_force_event { atributy }

Určuje vstupní impulsní sílu vloženou do objektu na určitý čas. V jednom objektu jich může být více. Atributy jsou následující:

- *vector* **force** – Určuje velikost vstupní síly.
- *real* **time** – určuje v jakém čase se síla projeví.

Následují objekty které jsou ve scéně viditelné.

plane

plane **<název>** { atributy }

Plocha o nekonečné velikosti, její atributy jsou:

- *vector* **normal** – normála plochy
- *vector* **position** – pozice plochy
- *material* **<název>**

sphere

sphere **<název>** { atributy }

Koule, její atributy jsou:

- *vector* **position** – pozice koule v prostoru
- *real* **radius** – poloměr koule
- *material* **<název>**
- *add_force_event*
- *real* **use_physics** – Určuje, zda bude objekt zahrnut do simulace (1) či ne (0).
- *vector* **v_e** – Vektor udávající otočení koule v prostoru (pro textury).
- *vector* **v_n** – Vektor udávající normálu otočení koule v prostoru (pro textury)

triangle

`triangle <název> { atributy }`

Trojúhelník, jeho atributy jsou:

- *vector* **vertex0, vertex1, vertex2** – body trojúhelníku
- *material* **<název>**

rectangle

`rectangle <název> { atributy }`

Obdélník, jeho atributy jsou:

- *vector* **vertex0, position1** – body jedné hrany obdélníku.
- *vector* **vertex2** – určuje bod, kterým bude procházet rovnověžná hrana k té definované.
- *material* <**název**>

box

`box <název> { atributy }`

Kvádr, jeho atributy jsou:

- *vector* **vertex0, vertex1** – Body jedné hrany základny(obdélníku).
- *vector* **vertex2** – Určuje bod, kterým bude procházet rovnověžná hrana základny k definované hraně základny.
- *real* **height** – Výška kvádru na kolmici k základně.
- **use_physics** - Určuje, zda bude objekt zahrnut do simulace (1) či ne (0).

- *material* <**název**>

Nastavení parametrů scény

`settings <název> { atributy }`

V tomto prvku se nastavuje počet vzorků pro jednostlivé efekty, dále velikost výstupního obrázku. Atributy tedy jsou:

- *real* **width** – šířka v pixelech výstupního obrázku.
- *real* **height** – výška v pixelech výstupního obrázku.
- *real* **fps** – počet snímků za sekundu pro všechny kamery ve scéně.
- *real* **interpolate samples**_ – počet vzorků interpolace pixelů rozptýlením paprsků pomocí „gaussovy“ matice.
- *real* **blur_samples** – počet vzorků efektu motion blur.
- *real* **blur_duration** – délka trvání otevření závěrky.
- *real* **blur_switch_ratio** – koeficient určující poměr změny závěrky ku otevřenému stavu.
- *real* **dof_samples** – počet vzorků efektu DOF.
- *real* **dof_c** – velikost kruhu rozptylu pro efekt DOF.
- *real* **soft_shadow_samples** – počet vzorků měkkých stínů.
- *real* **diffuse_reflection_samples** – počet vzorků pro matné povrchy.
- *real* **diffuse_refraction_samples** – počet vzorků pro průsvitné objekty.

Pokud některý z parametrů není zadán použije se výchozí a program o tom informuje uživatele pomocí textového výstupu.

Nastavení kamery

Kamer ve scéně může být více. Dobrým zvykem je řadit kamery podle toho kdy mají scénu vykreslovat, aby nedošlo k případným problémům. Pokud je někde přiřazena hodnota -1 porgram se pokusí použít implicitní nastavení.

`camera <název> { atributy }`

V tomto prvku se nachází komplexní nastavení projekční plochy, jeho natočení apod. Atributy jsou následující:

- *real* **video_start** – Počátek snímání této kamery v čase videa.
- *real* **video_end** – Konec snímání této kamery v čase videa.
- *real* **time_start** – Počátek snímání této kamery v čase simulované scény.
- *real* **time_end** – Konec snímání této kamery v čase simulované scény.
- *real* **DOF_start** – Vzdálenost, ve které se jeví objekty jako dokonale ostré (pro začátek času snímáné kamery).
- *real* **DOF_end** - Vzdálenost, ve které se jeví objekty jako dokonale ostré (pro začátek času snímáné kamery).
- *real* **FOV_start** – parametr určující úhel pohledu a tedy i přiblížení (pro začátek času snímáné kamery).
- *real* **FOV_end** - parametr určující úhel pohledu a tedy i přiblížení (pro začátek času snímáné kamery).
- *vector* **add_position** – přidá pozici kamery (může být více – interpoluje se)
- *vector* **add_target** – přidá pozici cíle kamery (může být více – interpoluje se)
- *vector* **sky_start** – udává „nebe kamery“, tedy rotaci kamery podle směru pohledu (pro začátek času snímané kamery).
- *vector* **sky_end** – udává „nebe kamery“, tedy rotaci kamery podle směru pohledu (pro začátek času snímané kamery).

FOV je číslo určující vzdálenost kamery od projekční plochy. Úhel výhledu lze vypočítat následovně:

úhel výhledu = 2 * arctan(FOV / 2).

Příklad nastavení scény

Jednoduchá scéna s jednou kulečníkovou koulí a jední kvádrem jako deska.

```
settings SCENA1 {
	width  640   height 480   fps 2   interpolate_samples 0   blur_samples 32
	blur_duration -1   blur_switch_ratio 0.2   dof_samples 18   dof_c 0.2
	soft_shadow_samples 2   diffuse_reflection_samples 2
	diffuse_refraction_samples 0   recursion 1
}

camera Camera1 {
	video_start 0   video_end 8   time_start 0   time_end 8
	add_position { 135 20 70 }   add_position { 50 20 70 }
	add_target { 120 4 55 }   add_target { 0 0 30 }
	DOF_start 20 DOF_end 60 FOV_start 2.5   FOV_end 6
	sky_start { 0 1 0 } sky_end { 0 1 0 }
}

phong_info BallRed {
	ambient { 0.6 0 0 } diffuse { 0.5 0 0 } specular { 0.7 0.7 0.7 }
	shininess 20 reflectance 0.2 diffuse_reflection 0.15 refraction 0
	refraction_index 1.3
}

phong_info BilliardTable {
	ambient { 0.2 0.4 0.25 } diffuse { 0.1 0.4 0.15 }
	specular { 0.05 0.05  0.05 } shininess 20 reflectance  0
	diffuse_reflection 0 refraction 0 refraction_index 1.3
}

texture BallRed     {
	filename textures/ball_red.tga
}

material_texture    BallRed {
	phong_info BallRed texture BallRed scale_U 0.5 scale_V 1
}

material_phong      BilliardTable{
	phong_info BilliardTable
}

sphere BallRed {
	position { 60 3.8 54.5 } radius 4 material BallRed
	v_e { 0 1 0 } v_n { 1 0 1 } use_physics 1
	add_force_event { force { 100000 0 0 } time 0.1 }
}

box ground {

	vertex0 { 0 -5 0 }  vertex1 { 200 -5 0  }  vertex2 { 200 -5 100  }
	heigth 5  material BilliardTable use_physics 1
}
```

Implementované prvky v raytraceru

- Phongův osvětlovací model
- Rekurzivní výpočty (odrazy, lomy)
- Distribuované výpočty
  - Interpolace paprsků do pixelu pomocí 2D gaussovy plochy (gauss x gauss).
  - Stínové paprsky (měkké stíny).
  - Odrazy (matné povrchy).
  - Refrakce (průsvitné materiály).
  - Primární paprsky (hloubka ostrosti).
  - Časová distribuce scény.
- Tvary: Koule, plocha, válec, trojúhelník, kosodélník (ze 2 trojúhelníků), obdélník (ze 2 trojúhelníků). Osově nezarovnaný kvádr (ze 6ti obdélníků). Některé z nich ovšem zatím nejdou načíst, protože nejsou implementované v Parseru.
- Světla bodové, obdélníkové a kulové.
- Materiály:
  - Natavení pro phongův model a pro distribuované efekty.
  - 2D textury (TGA bez komprese) na koulích, plochách, obdélnících.
    - S nastavením počátku textury pomocí 2 vektorů.
- Podpora načítání ze souboru
  - Kompletní nastavení parametrů výstupu a všech efektů.
  - Komplexní načítání materiálů, označení materiálu jménem a použití u objektů.
  - Načítání objektů se všemi parametry (pozice, tvar, natočení koulí), dále jestli mají být zahrnuty do simulace scény.
  - Pro objekty lze pro různé časy vložit vstupní impulsní hybnost.
  - Komplexní nastavení kamer.
- Fyzikální a časová simulace scény, interakce
  - Dopředná simulace scény
  - Každý objekt má své pole pozic v čase.
  - Interpolace neuložených časových z pozic uložených.
  - Fyzikální simulace koulí (s rotací), kvádrů (bez výpočtů rotací) a interakce mezi nimi.
  - Koulím lze udat v určitém čase určitou sílu určitým směrem.
- Komplexní nastavení kamer, scény a výstupu
  - Nastavení rozlišení obrazu (výstup BMP), interpolace, počet fps, délka a typ otevření závěrky.
  - Pro jednu scénu je možné mít více kamer.
  - Každá kamera má vlastnosti:
    - Počáteční a koncový čas ve videu.
    - Počáteční a koncový čas v simulované scéně (scéna může být zpomalená, zrychlená, pozpátku).
    - Více pozic jedné kamery, mezi kterými se pohybuje po beziérové křivce.
    - Nastavení více cílů jedné kamery, mezi kterými „prochází“ po beziérové křivce.
    - Počáteční a koncové zaměření hloubky ostrosti pro kameru (s plynulým přechodem). Je možné nechat nastavení na vzdálenosti cíle od kamery.
    - Počáteční a koncové nastavení úhlu výhledu (s plynulým přechodem).
    - Počáteční a koncové nastavení rotace kamery (s plynulým přechodem).

Seznam příloh

Příloha 1. Manuál

Příloha 2. DVD s programem, zdrojovými soubory, obrazovými výstupy, nastavením scén, texturami, dokumentací, a kompletním diagramem tříd.