Participating media, realtime metody

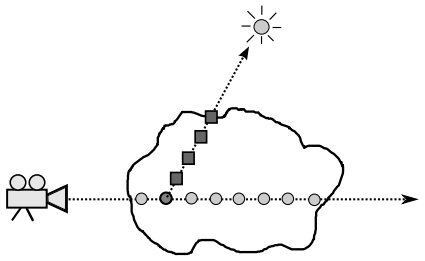
Zpracoval Martin Růžička, Mária Vámošová, 18. července 2012

Úvod

V 1. části jsme popsali Objemovou zobrazovací rovnici. Ta provádí výpočet radiance v daném místě přes celý objem scény a jde tak o řádově složitější výpočet v porovnání s klasickou Zobrazovací rovnicí. Vzhledem ke své velké výpočetní složitosti se proto interaktivní metody neobejdou bez různých předpokladů a zjednodušení.

Ray Marching

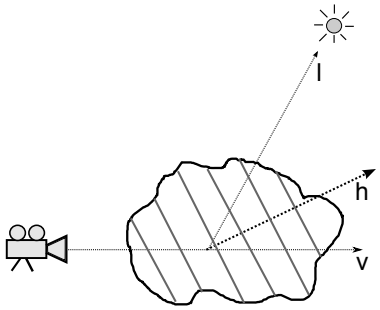
Je metoda založená na ray castingu. Od kamery vrháme ve směru pohledu paprsky do scény a hledáme první a poslední průsečík s médiem. Segment mezi průsečíky potom vzorkujeme. Na těchto vzorcích počítáme osvětlení. To provádíme vzorkováním vektoru, vedeného z pozice aktuálního vzorku směrem ke světelnému zdroji. Osvětlení spočtené na vzorcích segmentu nakonec použijeme k určení výsledné radiance směřující do kamery. Médium je obvykle reprezentováno formou 3D mřížky nebo analytickou funkcí. Výhodou algoritmu je především jednoduchost. K nevýhodám pak patří nízká rychlost, omezení na single scattering a problematický antialiasing.



Obrázek 1: Vzorkování paprsku procházejícího mediem je značeno kruhy. Čtverce znázorňují vzorkování light vektoru.

Slice based metody

Jsou metody podobné Ray Marchingu. V základní podobě slouží pouze k zobrazování média, nikoliv k jeho osvětlení a jak název napovídá, jedná se o metody, kdy médium prokládáme řezy. Roviny řezů jsou orientovány stejně jako obrazová rovina a jsou rozmístěny rovnoměrně po celém médiu. Každému řezu odpovídá "plátek media" (ta část media, kterou řez proniká). Tyto plátky nakonec kreslíme v pořadí odzadu směrem ke kameře. Popsaný algoritmus lze jednoduše implementovat na GPU. Nevýhodou je naopak tzv. Slicing artifact a fakt, že takto popsaný algoritmus nepočítá žádné osvětlení. Uvedenou metodu můžeme rozšířit o počítání propagace světla v médiu. Abychom to mohli udělat je potřeba nejprve změnit orientaci řezů. Řezy od teď budou kolmé na half vector mezi view a light vektorem. Řezy dale setřídíme vzestupně podle vzdálenosti od světelného zdroje. Tím dostaneme pořadí ve kterém (přibližně) probíhá propagace světla médiem. Můžeme tak počítat přenos světla mezi sousedními plátky a spočítat tím osvětlení v každém místě každého řezu. Ačkoliv není toto vylepšení zcela fyzikálně korektní, dává alespoň přibližný odhad osvětlení a významně tak zlepšuje celkový vizuální dojem.



Obrázek 2: **v** značí vektor vržený od kamery směrem do scény; **l** je vektor vedený z média ke světlu; **h** značí jejich half vektor a všechny řezy s ním svírají pravý úhel.

Billboarding metoda

Další metoda je založená na billboardech. Billboardy jsou polygony, které jsou orientované vždy směrem ke kameře. Těmito billboardy aproximujeme medium. Každý billboard reprezentuje objemovou část media. Obvykle je každému billboardu přirazena pozice, směr pohybu, životnost, textura, průhlednost a velikost, (=objem který reprezentuje). Billboardy seřadíme sestupně (nejbližší billboardy kreslíme poslední kvůli spravnému blendingu) podle vzdálenosti od kamery a vykreslíme. Tato metoda je rychlá, umožňuje animovat medium a lze ji jednoduše implementovat na GPU. Proto je s oblibou používaná v real-time grafice. Nevýhodou je hlavně problematické stínování částic, nízká přesnost aproximace a hranové artefakty. Hranové artefakty vznikají v místech, kde se billboardy protíná s geometrii scény. Popsaný problem řeší tzv. Soft particles. Ty modulují barvu

billboardu v závislosti na vzdálenosti billboardu a geometrie scény. Vzdálenost měříme pomocí hloubkové mapy generované z pohledu kamery.

Analytické metody

Za určitých podmínek můžeme propagaci světla médiem spočítat i analyticky. Jednu z takových metod publikoval například Sun a kol. Ta předpokládá homogenní medium rozprostírající se celou scénou, single scattering a použití izotropních bodových světel. Za uvedených podmínek potom můžeme spočítat radianci v libovolném místě scény. [1] Výhoda analytických metod je zřejmá – není potřeba nic integrovat, do vzorce stačí dosadit a máme výsledek.

Instant volume radiosity

Tato metoda je postavená na klasické Instant radiositě. Světelný zdroj emituje VPL-ka (virtual point light), která se šíří scénou. Při interakci VPL s médiem (tzv. scattering event) si informace o něm ukládáme (podobně jako třeba u Photon mappingu). Po vystřílení všech VPL můžeme konečně spočítat osvětlení v libovolném místě media. To provedeme součtem příspěvků od virtuálních zdrojů, se kterými zacházíme jako by se jednalo o "obyčejné" zdroje světla. Při výpočtu příspěvku VPL je např. nutné určit útlum světla mezi VPL a osvětleným bodem. Takto spočtené osvětlení je nestranné (unbiased), ale bude zatížené velkým množstvím šumu, který se projevuje výrazně zářícími body okolo každého VPL. Pro jejich odstranění je potřeba provést tzv. clamping, tj. oříznutí přízpěvku VPL na nejakou maximální hodnotu. Tím však ztratíme energii, kterou je potřeba kompenzovat. Kompenzaci lze provést např. pomocí sledování cest (path tracing), což by ale nevedlo na real-time řešení. Metoda proto zavádí řadu aproximací při výpočtu kompenzace, čímž přidáváme bias, který je pro nás však z vizualního hlediska přijatelnější než původní šum.

Cascaded light propagation

Je metoda určená primárně pro výpočet nízkofrekvenčního nepřímého osvětlení v plně dynamickém prostředí. Přímé osvětlení a stíny řešíme samostatně například pomocí lokálního osvětlovacího modelu a shadow mappingu. Hlavní datovou strukturou je zde 3D mřížka. Vlastní algoritmus můžeme rozdělit do 4 částí.

1. V prvním kroku vložíme do mřížky všechny zdroje nepřímého osvětlení. Zde postupujeme podobně jako v případě klasické instant radiosity. Střílení VPL ze světelných zdrojů však nahradíme pomocí RSM (reflective shadow map). RSM generujeme z pozice světel a každý texel této stínové mapy reprezentuje jedno vystřelené VPL. Intenzitu a směr záření jednotlivých VPL z důvodů pameťové úspory reprezentujeme pomocí SH (spherical harmonics)

[1] Kdo se ten šílený vzorec chce učit ať si ho vyhledá ve slidech .)

2. V druhé fázi do 3D mřížky přidáme geometrii scény (opět pomocí RSM, nebo případně vzorkováním geometrie scény). Z těchto informací spočteme faktor zakrytí jednotlivých buňek v mřížce a tuto informaci si ke každé buňce uložíme.

3. Vlastní propagace světla mřížkou probíhá iterativně, kdy "předáváme" tok (flux) mezi sousedními buňkami. Množství toku přenesené mezi sousedními buňkami závisí na faktoru viditelnosti těchto buněk, jejich vzájemné orientaci a velikosti záře, kterou mezi buňkami přenášíme. Množství přijatého toku si pro jednotlivé buňky ukládáme opět ve formě SH.

4. V závěrečném kroku spočteme z informací uložených v buňkách výsledné osvětlení scény. Vzhledem ke své povaze lze uvedený algoritmus dobře rozšířit o interakci s objemovým médiem. Upravit bychom museli 1. a 3. krok, které rozšíříme o interakci s médiem.

Deep Shadow Maps

Normálna shadow mapa ukladá jednu hodnotu na pixel: hĺbku najbližšieho objektu. Nie je v nej napríklad uložená informácia o transparentnosti.

Deep shadow map je rozšírenie shadow mapy tak, že pre každý pixel sa uloží nie jedna hodnota, ale celá funkcia viditeľnosti pozdĺž paprsku. Funkcia je závislá na hodnote hĺbky z, na začiatku má funkcia hodnotu 1, a postupne klesá - úplne zatienená oblasť má hodnotu 0. Zatienenie je spôsobené: polopriehľadnými objektmi, čiastočným zatienením pixelu solídnym objektom, a transmitanciou média.

Mapa sa vytvorí následovne: Najprv si spočítame priesečníky s objektami, a z nich *surface transmittance function* - po častiach konštantnú funkciu danú priehľadnosťou objektov a čiastočným zakrytím pixelu.

Z rovnomerného nasamplovania v médiu dostanem *extinction function* $\kappa(z)$. Táto funkcia vznikne lineárnou interpoláciou medzi *extinction coefficients* κ_i v daných samploch v hĺbke z_i, ktoré vyjadrujú úbytok svetla pozdĺž paprsku na jednotku vzdialenosti. Podiel svetla, ktoré dopadne do daného bodu potom dostaneme ako

$$\tau^v(z) = exp(-\int_0^z \kappa(z')dz')$$

Kedže táto funkcia nie je po častiach lineárna, aproximujeme ju tým, že vyhodnotíme transmittance pre každý lineárny úsek ako

$$T_i = exp(-(z^v_{i+1} - z^v_i)(\kappa_{i+1} + \kappa_i)/2)$$

kde z^v_i je hĺbka volumetrického vzorku i a κ_i jej *extinction coefficient*.

Upravenú *volume transmittance function* τ^v potom dostaneme vynásobením T_i pre jednotlivé segmenty.

Funkciu celkovej transmitancie potom dostaneme vynásobením týchto dvoch funkcií. Názorný postup na obrázku. [obrázok 3, pôvodný Figure 4.]

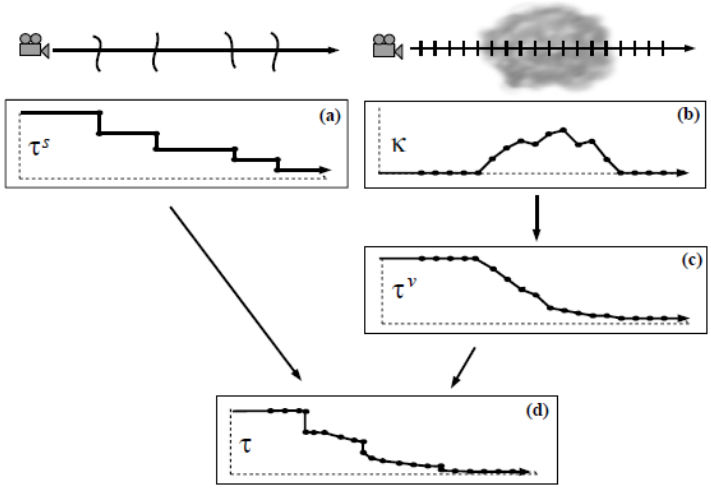


Figure 4: *Constructing a transmittance function.* ***(a)*** *The object intersections along a given ray yield the surface transmittance function τ^s, which has a discontinuity at the depth of each surface.* ***(b)*** *The extinction function κ is obtained by sampling the atmospheric density at regular intervals along the ray.* ***(c)*** *The extinction function is integrated and exponentiated to yield the volume transmittance τ^v.* ***(d)*** *The surface transmittance and volume transmittance are multiplied to obtain the final transmittance function τ for each ray.*

Obrázek 3: Skonštruovanie funkcie transmitancie

Keďže týmto postupom je vygenerovaná funkcia definovaná veľkým množstvom bodov, chceme ju nejak zakomprimovať - aby výsledná funkcia bola čo najpodobnejšia, ale stačilo výrazne menej bodov na jej reprezentáciu.

Myšlienka kompresie je taká, že hodnota po kompresii $\tau_{i'}$ sa môže líšiť od τ_i o ϵ. Okolo každej hodnoty teda vznikne rozsah možných hodnôt. Kedže chceme, aby výsledná funkcia bola po častiach lineárna, pre každý bod dostaneme intervaly pre možné lineárne funkcie prechádzajúce povolenými hodnotami. Prienikom týchto intervalov nájdeme funkciu ktorá reprezentuje viacero susedných bodov. Ak interval susedného bodu nemá spoločný prienik, tak odtiaľ začnem nový úsek. Najlepšie je to vidno na obrázku [obrázok 4: pôvodný Figure 5]

Na GPU sa deep shadow mapy dajú implementovať použitím viacerých textúr v rôznych rovinách.

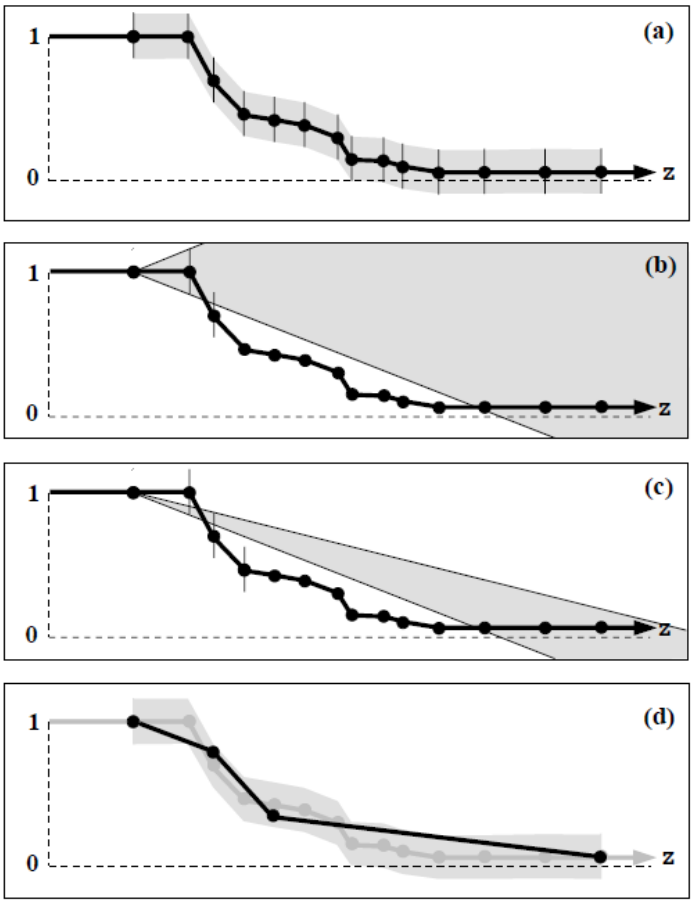


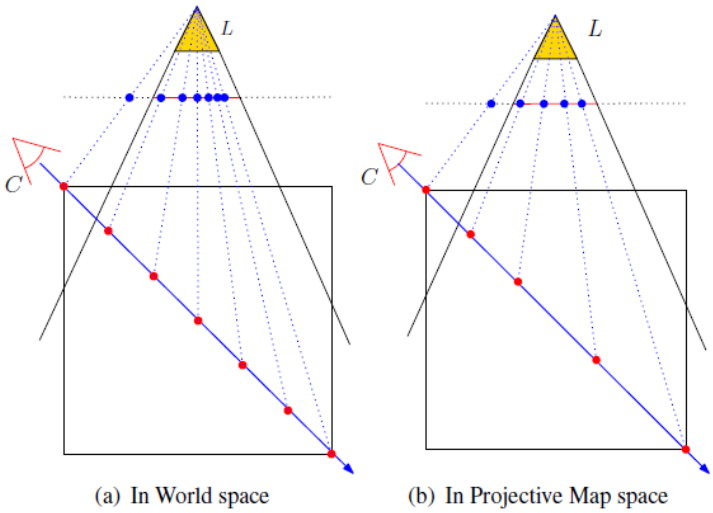
Figure 5: *Our compression algorithm.* ***(a)*** *A piecewise linear curve and an illustration of its error bound.* ***(b)*** *Each input vertex defines a* target window *that constrains the slope of the next output segment.* ***(c)*** *The current slope range is intersected with each target window until it would become empty.* ***(d)*** *The output segment is extended to the current z value with a slope equal to the midpoint of the current slope range, and this process is repeated.*

Obrázek 4: Kompresia

Tiene na médiu na GPU

Máme homogénne médium a chceme vyrendrovať scénu s tieňmi na médiu, na GPU. Vytvoríme si shadow mapu klasickým spôsobom, vyrendrovaním scény z pohľadu svetla. Túto shadow mapu potom používame pri ray-marchingu pri rendrovaní média. Ak budeme pri ray-marchingu vzorkovať uniformne, po premietnutí vzorkov na shadow mapu budú dotazy do shadow mapy neuniformné. Riešením je vzorkovať uniformne v shadow mape, a premietať tieto vzorky na paprsok [obrázok 5]

V homogénnom médiu môžeme pre utlmenie pozdĺž paprsku použiť analytické vyjadrenie. V nehomogénnom médiu použijeme Deep Shadow Mapy, ktoré si ale musíme nejak uložiť. Klasické deep shadow mapy sú po častiach lineárne funkcie reprezentované hodnotami v určitých bodoch, a pre každý texel môže byť počet bodov rôzny. Pre GPU je vhodnejšie, ak by sme Deep Shadow mapu reprezento-



(a) In World space (b) In Projective Map space

Obrázek 5:
(a) Uniformné vzorkovanie pozdĺž paprsku.
(b) Uniformné vzorkovanie v shadow mape, premietané na paprsok.

vali nejak inak, s menším počtom koeficientov, ktorý by bol rovnaký pre každý texel. To vyriešime tak, že na pôvodnej shadow mape urobíme diskrétnu kosínovú transformáciu, a vezmeme prvých 16 koeficientov. Tá síce zle aproximuje skokovú funkciu čiastočného zakrytia a polopriehľadnosti, ale na tiene z média samotného je to po regularizácii okej.

Zobrazování atmosféry

Atmosféra je jedním z nejobvyklejších médií s kterým se v běžném životě setkáváme. Proto se na něj podíváme v této části podrobněji. Atmosféra obklopuje celou naši Zemi, je to velké ale řídké médium. Narozdíl od mnohých jiných médií má malou absorbci. To je pro nás z hlediska renderingu nepříjemné protože nemůžeme výpočet šířícího se záření předčasně ukončit například pomocí Ruské rulety. Výhodou je pro nás naopak stabilita atmosféry z hlediska osvětlení (slunce se hýbe jen pomalu) a hustoty. Hustota atmosféry je přes den relativně stálá a s rostoucí nadmořskou výškou klesá exponencielně (každých cca 5.5 km se její hustota zmenší o polovinu) a od určité nadmořské výšky ji tak můžeme začít ignorovat. Z obecných algoritmů lze pro výpočet radiance v atmosféře použít Path tracing i Volumetric radiance trasfer (=objemová radiozita). Photon mapping se zde naopak typicky příliš nepoužívá, protože z důvodu velikého rozsahu atmosféry bychom pro reprezentaci osvětlení potřebovali obrovské množství fotonů. V následující části si uvedeme několik specializovaných algoritmů.

Preetham´s model

Je analytická a empirická metoda pro výpočet atmosférického osvětlení. Vzorec pro výpočet luminance oblohy je uveden níže.

$$Y = Y_z \frac{F(\theta, \gamma)}{F(0, \theta_S)}$$

- $F(\theta, \gamma) = (1 + Ae^{\frac{B}{\cos\theta}})(1 + Ce^{D\gamma} + E\cos^2\gamma)$
- θ_s je úhel mezi sluncem a zenitem
- θ je úhel mezi view vektorem a zenitem
- γ je úhel mezi sluncem a view vektorem
- A, B, C, D, E jsou lineární funkce turbidicity T, která udává zastíněnost oblohy ($T \in \langle 1, 10\rangle$, kde dokonale jasná obloha má $T = 1$)

Hlavní výhodou metody je jednoduchost a rychlost. Nevýhodou je omezení pouze na zemskou atmosféru, nulovou výšku pozorovatele a oblohu bez mraků.

Precomputed scattering

Atmosféru lze dobře parametrizovat pomocí několika málo atributů (například pozicí pozorovatele, směru jeho pohledu a pozice slunce). Precomputed scattering tímto způsobem předpočítá single scattering pro všechny přípustné kombinace parametrů a uloží je do n-dimenzionální tabulky, kde n odpovídá počtu použitých parametrů (typicky 4). Tento předvýpočet je neinteraktivní. Hodnoty z tabulky poté můžeme jednoduše a rychle používat. Pomocí dynamického programování navíc můžeme dopočítat i multiple scattering: $(k + 1)$-krát odražené světlo odpovídá spočtení single scatteringu z k-krát odraženého světla.

Zobrazování oblak

Oblaka mají v porovnání s atmosférou několik nepříjemných vlastností a je tak obtížnější je renderovat. Hlavním problémem je jejich vysoké albedo a silně anizotropická fázová funkce, která nám prakticky znemožňuje použití standardních algoritmů pro výpočet osvětlení jako je path tracing nebo volume radiance transfer. My si zde uvedeme 2 empirické modely pro jejich kreslení.

Wang

Wang publikovala metodu využívající billboardy. Objem mraku je definován pomocí předpřipraveného modelu. Do míst definovaných modelem poté rozmístíme billboardy s texturou mraku a ty zobrazíme. Takto vytvořený mrak tak zachovává svůj objem, vypadá poměrně slušně a přitom ho lze jednoduše a rychle zobrazovat. Jde o čistě empirický přístup použitý pro zobrazovaní mraků v Microsoft flight simulatoru.

Szirmay-Kalos

Mrak reprezentujeme N náhodně rozmístěnými částicemi a obdobně diskretizujeme i možné směry šíření světla. Dále předpokládáme, že k odrazům světla může docházet pouze v místech těchto částic mraku. Mezi těmito částicemi potom iterativně distribuujeme světlo a počítáme jeho celkové množsví v jednotlivých částicích mraku. Nakonec osvětlené částice reprezentující mrak zobrazíme.