VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

PROCEDURÁLNĚ GENEROVANÁ KRAJINA VE FRAGMENT SHADERU

PROCEDURALLY GENERATED LANDSCAPE IN FRAGMENT SHADER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. DENIS LEITNER**
AUTHOR

VEDOUCÍ PRÁCE **TOMÁŠ CHLUBNA, Ing.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání diplomové práce

Student: **Leitner Denis, Bc.**
Program: Informační technologie
Obor: Počítačová grafika a interakce
Název: **Procedurálně generovaná krajina ve fragment shaderu**
Procedurally Generated Landscape in Fragment Shader
Kategorie: Počítačová grafika
Zadání:

1. Seznamte se s konceptem vykreslovacího řetězce v OpenGL.
2. Nastudujte metody vhodné pro procedurální generování a vykreslování 3D obsahu za pomocí fragment shaderu bez vstupní geometrie (raymarching, SDF apod.).
3. Navrhněte netriviální scénu vhodnou pro demonstraci.
4. Implementujte navržené demo.
5. Zdokumentujte výsledek a vytvořte demonstrační video.

Literatura:

- Freiknecht, J.; Effelsberg, W. A Survey on the Procedural Generation of Virtual Worlds. Multimodal Technol. Interact. 2017, 1, 27. https://doi.org/10.3390/mti1040027
- Ebert, David S., et al. Texturing & modeling: a procedural approach. Academic Press, 2014. ISBN 1483297020, 9781483297026
- Osher, Stanley, en Ronald Fedkiw. "Signed Distance Functions". Level Set Methods and Dynamic Implicit Surfaces. New York, NY: Springer New York, 2003. 17-22.
- Tomczak, Lukasz Jaroslaw. "GPU Ray Marching of Distance Fields." *Technical University of Denmark* 8 (2012), Master's thesis.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3, experimenty vedoucí k bodu 4.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Chlubna Tomáš, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 18. května 2022
Datum schválení: 1. listopadu 2021

Abstrakt

Táto práca je zameraná na vykresľovanie procedurálne generovanej krajiny bez použitia vstupnej geometrie a textúr. Sú v nej popísané techniky používané na realistické vykresľovanie a generovanie prírodných scenérií, ktoré zahŕňajú generovanie terénu a realistické vykresľovanie atmosféry a oblačnosti. Práca ďalej popisuje vykresľovanie terénu a tieňov pomocou raymarchingu, rovnako ako aj realistické osvetlenie terénu s aproximáciou ambientného a nepriameho osvetlenia.

Abstract

This thesis deals with rendering of procedurally generated landscape without the use of input geometry or textures. It describes techniques for generation and realistic rendering of natural outdoor scenes. These techniques include terrain generation and realistic atmosphere and cloud rendering. Thesis also describes the use of raymarching for terrain and shadow rendering and realistic lighting for terrain which includes ambient and indirect light approximation.

Kľúčové slová

procedurálne generovaná grafika, procedurálne generovaný terén, procedurálne textúry, šum, fraktál, raymarching, SDF, atmosféra, rozptyl svetla, Rayleighov rozptyl, Mieho rozptyl, oblaky, volumetrické vykresľovanie, fragment shader

Keywords

procedurally generated graphics, procedurally generated terrain, procedural textures, noise, fractal, raymarching, SDF, atmosphere, light scattering, Rayleigh scattering, Mie scattering, clouds, volumetric rendering, fragment shader

Citácia

LEITNER, Denis. *Procedurálně generovaná krajina ve fragment shaderu.* Brno, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Tomáš Chlubna, Ing.

Procedurálně generovaná krajina ve fragment shaderu

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána Ing. Tomáša Chlubnu. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Denis Leitner
18. mája 2022

Poďakovanie

Rád by som sa poďakoval pánovi Ing. Tomášovi Chlubnovi za vedenie tejto práce a poskytnuté rady.

Obsah

1 Úvod **2**

2 Procedurálne generovaná grafika **3**
2.1 Techniky procedurálneho generovania grafiky 3
2.2 Fraktálna geometria 4
2.3 Procedurálne generovaný terén 7

3 Realistické zobrazovanie krajiny **12**
3.1 Šírenie svetla v médiu 12
3.2 Simulácia atmosféry 20
3.3 Vykresľovanie procedurálnej výškovej mapy terénu 24

4 Návrh riešenia **27**
4.1 Vykresľovací reťazec 27
4.2 Terén 28
4.3 Atmosféra 32
4.4 Oblaky 35

5 Implementácia **37**
5.1 Vykresľovanie vo fragment shaderi 37
5.2 Implementovaná scéna 37
5.3 Generovanie šumu 38
5.4 Výpočet normál terénu 39
5.5 Textúrovanie 39
5.6 Ovládanie 40

6 Vyhodnotenie **42**
6.1 Porovnanie grafických nastavení 42
6.2 Vplyv počtu krokov raymarchingu na výkon 43
6.3 Vplyv počtu krokov raymarchingu na výsledný obraz 44

7 Záver **47**

Literatúra **48**

Kapitola 1

Úvod

Cieľom tejto práce je vytvorenie grafického dema na vykresľovanie procedurálne generovanej krajiny vo fragment shaderi. Znamená to, že celý obsah, ako je napríklad terén, textúry a obloha, musí byť generovaný vo fragment shaderi bez použitia vstupných modelov a textúr. Veľký dôraz je rovnako kladený aj na realistické zobrazenie generovaného obsahu.

Realistické zobrazovanie krajiny je zaujímavá ale aj zložitá úlohá. Jej riešenie má uplatnenie v mnohých odvetviach, napríklad vo filmovom priemysle, architektonickej vizualizácii alebo počítačových hrách. Pri vykresľovaní krajiny je potrebné okrem terénu kvalitne zobraziť aj volumetrické javy, ktoré sú na dosiahnutie realizmu vo vonkajších scénach veľmi dôležité. Volumetrické javy, ktorými sa zaoberá táto práca sú atmosféra a oblaky.

Postupy na generovanie procedurálnej grafiky sú popísané v kapitole 2. V tejto kapitole sú zjednodušene vysvetlené fraktály, pretože práve pomocou nich sa najlepšie popisujú a generujú zložité prírodné útvary, ako napríklad terén a oblaky. Táto kapitola obsahuje tri rôzne algoritmy na generovanie terénu.

V kapitole 3 je popísané všetko okolo realistického zobrazovania krajiny. Veľká časť kapitoly je zameraná na popis šírenia svetla v médiu a na volumetrické vykresľovanie, ktorým je zobrazovaná atmosféra a oblaky. Na konci kapitoly je predstavený algoritmus na vykresľovanie terénu.

Kapitola 4 je zameraná na návrh riešenia. Je v nej popísaný vykresľovací reťazec a spôsob riešenia všetkých častí vykresľovania.

V kapitole 5 sú popísané niektoré zaujímavé implementačné detaily. V kapitole 6 sú popísané vykonané merania a aj porovnané výstupy pre rôzne parametre vykresľovania.

Kapitola 2

Procedurálne generovaná grafika

Táto kapitola popisuje úvod do procedurálne generovanej grafiky. Jej hlavným cieľom je však predstaviť techniky a algoritmy, ktoré sa používajú pri procedurálnom generovaní terénu. Keďže tieto techniky vo veľkej miere využívajú znalosti z fraktálnej geometrie, táto kapitola obsahuje aj krátky úvod do fraktálov.

2.1 Techniky procedurálneho generovania grafiky

Za techniky procedurálneho generovania grafiky sa považujú časti kódu alebo algoritmy, ktoré špecifikujú isté charakteristiky počítačovo-generovaného modelu alebo efektu [3]. Tieto techniky nepoužívajú vstupné obrázky na popis farby textúry a ani 3D modely na popis geometrie objektu. Miesto toho sa textúry a geometria generujú pomocou algoritmov a matematických funkcií. Procedurálne techniky sa využívajú v mnohých oblastiach počítačovej grafiky, od počítačových hier až po filmy.

Prvé procedurálne techniky boli vytvorené na generovanie textúr. Ken Perlin, Darwyn Peachey a Geoffrey Gardner predstavili v roku 1985 techniky 3D textúrovania, čo spôsobilo veľký záujem o procedurálne generovanú grafiku [3]. Tieto techniky umožnili vytvorenie prvých realistických textúr mramoru, dreva, kameňa a oblakov. Procedurálne techniky sa ďalej vyvíjali a v dnešnej dobe sa používajú okrem textúrovania a modelovania aj na tvorbu procedurálnych animácií.

2.1.1 Výhody a nevýhody procedurálne generovanej grafiky

Najväčšia výhoda generovania grafiky pomocou procedurálnych techník je minimalizácia využitia pamäte. Namiesto ukladania rozsiahlych dát scény alebo animácie do pamäte sa tieto dáta abstrahujú do procedúry (funkcie alebo algoritmu), ktorá sa vyhodnotí keď sú dané dáta potrebné. Týmto sa šetrí miesto v pamäti a takisto sa aj eliminuje prístup do pamäte, ktorý môže byť oproti výpočtu často pomalší. Keďže dáta nie sú explicitne definované, procedurálne generovanie umožní vytvorenie textúr a modelov s práve potrebným rozlíšením a úrovňou detailu.

Odborník na procedurálnu grafiku David Ebert [3] považuje za veľkú výhodu možnosť parametrizácie týchto procedúr, čo umožní priradiť parametru zmysluplný význam (napr. parameter, ktorý mení členitosť terénu). Používateľ dokáže zmenou hodnôt parametrov ovplyvniť výstup procedúry a tým jednoducho prispôsobiť výstup svojim potrebám. Podľa Eberta parametrizácia odbremení používateľa od potreby nízkoúrovňovej špecifikácie detai-

lov. Procedurálne techniky dokážu transformovať niekoľko vstupných parametrov na veľmi komplexný a detailný výstup, čo šetrí čas používateľa.

Abstrakcia dát do procedúry, ktorá bola vyššie popísaná ako výhoda, môže byť v niektorých prípadoch braná aj ako nevýhoda. Ak je procedúra príliš zložitá na výpočet alebo ju je potrebné vyhodnocovať veľmi často, takýto prístup sa stáva veľmi pomalý a obmedzuje výkon aplikácie. Zložité procedúry rovnako zvyšujú zložitosť kódu, čo má za následok napríklad dlhšie časy kompilácie shaderov.

2.1.2 Príklady procedurálnych techník

Procedurálne techniky na tvorbu grafiky sa používajú najčastejšie na generovanie prírodných objektov, textúr a javov. Na lepšiu predstavu o použití procedurálnych techník je nižšie predstavených niekoľko príkladov:

- **Modely založené na gramatikách** – Patria sem prevažne graftály a L-systémy. Tieto techniky umožňujú simulovať komplexnú štruktúru stromov, rastlín a iných prírodných objektov. Na popis pravidel rastu týchto objektov sa používajú formálne jazyky.
- **Fraktálna geometria** – Slúži napríklad na generovanie krajiny, kameňov, vodnej hladiny, koralov, prírodných textúr atď. Fraktálna geometria je bližšie popísaná v kapitole 2.2.
- **Implicitné plochy** – Používajú sa na modelovanie organických alebo iných zložitých a mäkkých objektov, ktoré by bolo komplikované animovať alebo popísať pomocou tradičných spôsobov [3]. Implicitné plochy sú plochy s konštantnou hodnotou, tzv. izoplochy, ktoré sú reprezentované implicitnou rovnicou v tvare $F(x, y, z) = 0$. Zložité modely je možné vytvoriť skombinovaním niekoľkých základných implicitných plôch. Geometrický tvar implicitných plôch nie je definovaný 3D modelárom/animátorom ale vyhodnotením implicitných funkcií, ktorými sú implicitné plochy definované.
- **Časticové systémy** – Používajú sa prevažne na generovanie explózií, simuláciu kŕdľa vtákov, ohňa, dymu atď. Časticový systém je reprezentovaný veľkou kolekciou jednoduchých geometrických častíc, ktoré sa časom menia. Správanie týchto častíc, konkrétne ich animácia, pozícia, vznik a zánik, sú kontrolované procedurálne.

2.2 Fraktálna geometria

Fraktálna geometria je vedná disciplína, ktorej základy boli predstavené v roku 1975 Benoitom B. Mandelbrotom v eseji *Les Objets Fractals: Forme, Hasard et Dimension* [10]. Fraktálna geometria predstavuje jednoduchý spôsob popisu komplexných tvarov, obzvlášť tvarov nachádzajúcich sa v prírode, ktoré by bolo zložité popísať euklidovskou geometriou.

Na rozdiel od euklidovskej geometrie, ktorá je väčšinou popísaná rovnicami, je fraktálna geometria popísaná algoritmom, najčastejšie rekurzívnym [20]. Z tohoto dôvodu je výskum a vývoj fraktálov úzko spojený s počítačovou grafikou. Princípy fraktálnej geometrie boli matematikom známe už nejaký čas pred Mandelbrotom, ale až vývoj počítačov a počítačovej grafiky umožnil ich bližšie skúmanie.

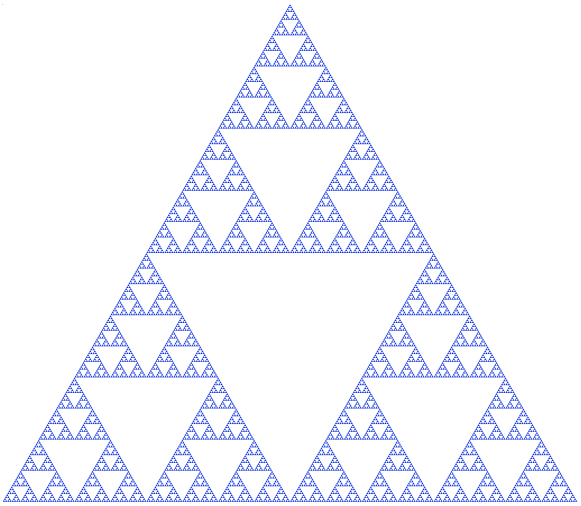
2.2.1 Fraktál

Fraktál môže byť definovaný ako geometricky komplexný objekt, ktorého komplexnosť vzniká v dôsledku opakovania určitého tvaru v rôznych veľkostiach [3]. Jednou z hlavných charakteristických vlastností fraktálov je *sebepodobnosť (angl. self-similarity)*. Táto vlastnosť znamená, že fraktál je invariantný voči zmene mierky – dá sa rozložiť na menšie časti, z ktorých každá je zmenšenou kópiou originálneho fraktálu [20]. Navyše každá časť sa dá ďalej rozložiť na ešte menšie časti, ktoré sú zmenšené kópie pôvodnej časti a aj celého fraktálu. Tieto kópie môžu byť presne alebo iba približne rovnaké. Z tohto hľadiska sa rozlišujú dva druhy sebepodobnosti:

1. **Presná sebepodobnosť -** Menšie časti, na ktoré sa dá fraktál rozdeliť sú presné kópie celého fraktálu. Tieto fraktály sa nazývajú *deterministické*. Presne sebepodobný fraktál je napríklad Kochova krivka alebo Sierpińského trojuholník (viď obrázok 2.1).

2. **Štatistická sebepodobnosť -** Menšie časti fraktálu sú len približne (štatisticky) podobné pôvodnému fraktálu. Tento typ sebepodobnosti sa vo veľkej miere vyskytuje v prírode. Pozorovať ju je možné napríklad v štruktúre stromov (viď obrázok 2.2), pohorí, oblakov, riečnych systémov, bleskov, a mnohých iných prírodných útvarov. Všetky tieto útvary sú fraktálne.

   Fraktály vyznačujúce sa štatistickou sebepodobnosťou sa nazývajú *nedeterministické* alebo *náhodné fraktály*. Pri ich generovaní v počítačovej grafike sa používajú náhodné čísla a táto náhodnosť spôsobuje, že vygenerované útvary sú podobné prírodným formám. Spôsob ich generovania a ich využitie v počítačovej grafike je bližšie popísané nižšie, keďže tento typ fraktálov sa používa pri procedurálnom generovaní krajiny.



Obr. 2.1: Sierpińského trojuholník. Príklad deterministického fraktálu.[1]

2.2.2 Zlomkový Brownov Pohyb

Prírodné javy a tvary objektov nachádzajúcich sa v prírode sú ovplyvnené náhodnými procesmi [20]. Túto náhodnosť prírody je v počítačovej grafike možné modelovať pomocou

[1] Obrázok prevzatý z https://commons.wikimedia.org/wiki/File:Sierpinski_triangle.svg

Obr. 2.2: Stromy sú príklad fraktálnej štruktúry v prírode. Menšie vetvy sú podobné väčším vetvám a aj celému stromu.[2]

nedeterministických fraktálov. Jeden z najpoužívanejších nedeterministických fraktálov je *zlomkový Brownov pohyb*, ďalej označovaný *fBm (fractional Brownian motion)*. Používa sa napríklad na generovanie terénu, textúr, oblakov alebo na distribúciu stromov a iných prírodných objektov.

Hlavná myšlienka *fBm* je vytvorenie deterministickej hladkej náhodnosti pomocou nejakého šumu (môže byť gradientný, hodnotový, voronoise, ...), ktorým je vytvorená sebepodobnosť [18]. Táto sebepodobnosť *fBm* sa vytvorí opakovaním daného šumu v rôznych veľkostiach (frekvenciách). Ako je možné vidieť v algoritme 1, postup na vytvorene *fBm* je veľmi jednoduchý ale pri dostatočnom počte iterácií vytvorí výstup s veľkou mierou detailov.

Algoritmus 1: Algoritmus na generovanie zlomkového Brownovho pohybu [18].

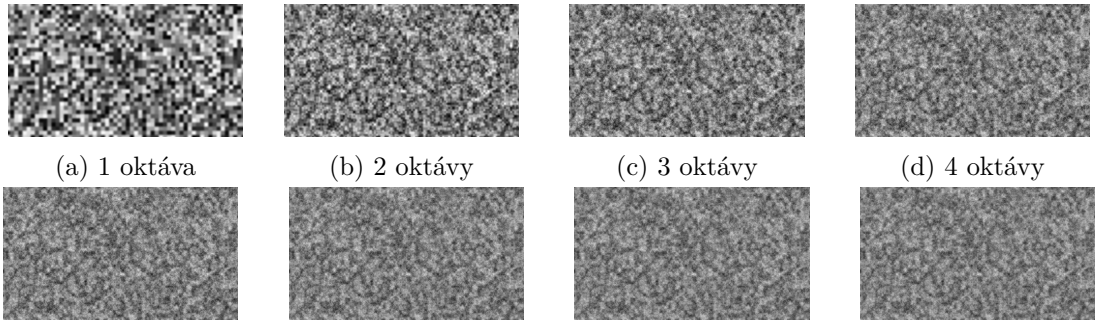
Input: Input point *point*, Hurst exponent H, number of octaves *octaves*
Output: Value of fBm in a given input point

```
1 result ← 0
2 for i = 0 to (octaves − 1) do
3 |   f ← 2^i                          // Frequency of the noise
4 |   a ← f^(−H)                       // Amplitude of the noise
5 |   signal ← a · Noise(f · point)
6 |   result ← result + signal
7 end
8 return result
```

Z algoritmu 1 je zjavné, že sa jedná o fraktál – *fBm* vzniká opakovaním určitého tvaru, v tomto prípade šumu (funkcia $Noise()$), v rôznych veľkostiach. V prvej iterácii sa začne so základným šumom, ku ktorému sa v každej ďalšej iterácii pridáva šum s dvojnásobnou frekvenciou a exponenciálne zníženou amplitúdou oproti predchádzajúcej iterácii. Parameter *octaves* určuje počet frekvencií, ktoré sú zlúčené do výsledku. Názov tohto parametru je prevzatý z hudby, kde dva tóny vzdialené o oktávu majú voči sebe dvakrát väčšiu/menšiu frekvenciu [18]. Väčší počet oktáv zabezpečí viac detailov, ale aj tu platí Nyquistov-Shannonov vzorkovací teorém, takže od určitého množstva oktáv nie je vidieť rozdiel v de-

[2]Obrázok prevzatý z http://www.cindyjgomez.com/fractaltrees/

tailoch. Vysoký počet oktáv je rovnako problematický pri použití *fBm* v 3D scénach, kde môže dôjsť k aliasingu. Na obrázku 2.3 sú porovnané textúry vygenerované pomocou *fBm* s rôznym počtom oktáv.

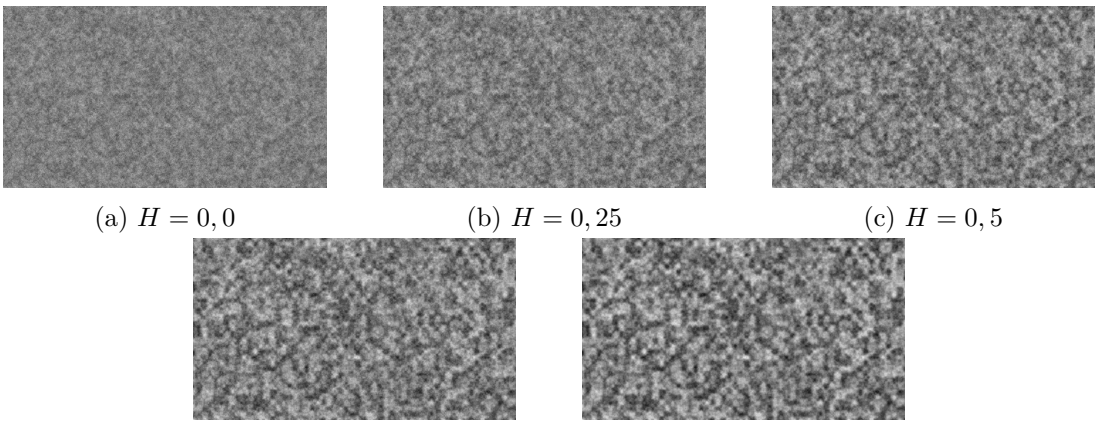


(a) 1 oktáva (b) 2 oktávy (c) 3 oktávy (d) 4 oktávy

(e) 5 oktáv (f) 6 oktáv (g) 7 oktáv (h) 8 oktáv

Obr. 2.3: Porovnanie *fBm* pre rôzny počet oktáv šumu.

Parameter H z algoritmu 1 sa nazýva *Hurstov exponent* a tento parameter ovplyvňuje členitosť výsledného fraktálu. Pre $H = 1$ je výsledok pomerne hladký, ale ako sa H približuje k nule, výsledok je viac členitý a začína sa podobať bielemu šumu (viď obrázok 2.4).



(a) $H = 0,0$ (b) $H = 0,25$ (c) $H = 0,5$

(d) $H = 0,75$ (e) $H = 1,0$

Obr. 2.4: Porovnanie *fBm* s rôznymi hodnotami Hurstovho exponentu.

2.3 Procedurálne generovaný terén

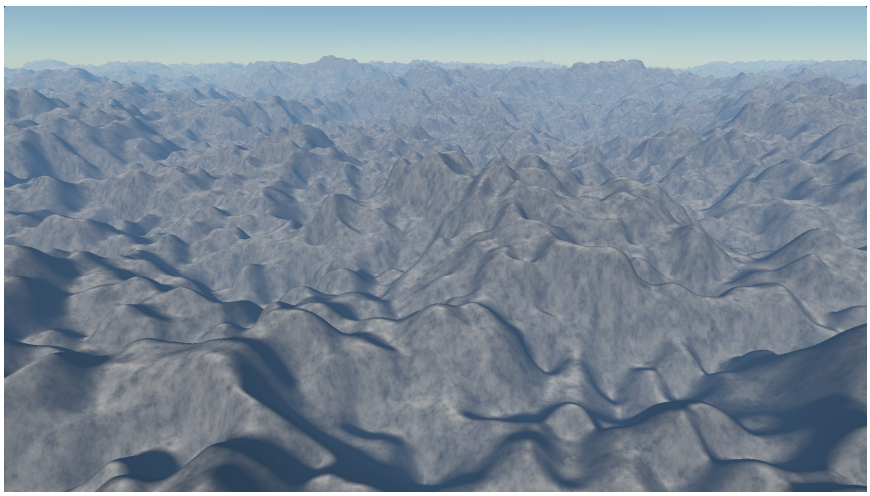
Modely terénu bývajú v počítačovej grafike zvyčajne reprezentované výškovou mapou. Výšková mapa je dvojrozmerné pole nadmorských výšok v pravidelných intervaloch [3]. Každý prvok tohto poľa obsahuje len jednu hodnotu, z čoho vyplýva, že útvary ako jaskyne alebo skalné previsy nie je možné reprezentovať výškovou mapou. Výškovú mapu je možné získať rôznymi spôsobmi, napríklad:

- 3D modelovanie
- aplikácie na tvorbu terénu

- reálne dáta
- procedurálne generovanie

Tento text sa bude zaoberať len poslednou metódou – procedurálnym generovaním. Procedurálne generovaný terén sa používa predovšetkým v počítačových hrách a filmoch. Najznámejšie počítačové hry s procedurálne generovanými svetmi sú napríklad *Minecraft*[3] alebo *No Man's Sky*[4]. Hráči dokážu hrať tieto hry stovky hodín, pretože vždy objavujú niečo nové a nachádzajú svety a terény, ktoré pred nimi ešte nikto iný nenavštívil [12]. Takúto rôznorodosť a komplexnosť je možné zostrojiť len pomocou nedeterministických fraktálov.

Prvé procedurálne generované terény boli tvorené pomocou *fBm*, ktorý bol popísaný vyššie. Takto generované terény sa podobajú na pohoria. Ich problém je však, že v každom mieste majú rovnakú členitosť (sú homogénne), čo úplne neodpovedá prírode. V prírode je terén v rôznych miestach rôzne členitý – vyššie časti pohoria sú viac členité ako nižšie časti pohoria. V teréne tvoreného pomocou *fBm* sa rovnako nedajú nájsť útvary ako napríklad roviny alebo menšie menej členité kopce, ktoré postupne prechádzajú do vyššieho skalnatého pohoria. Príklad terénu vygenerovaného pomocou *fBm* je zobrazený na obrázku 2.5.



Obr. 2.5: Fraktálny terén vygenerovaný pomocou *fBm* so šiestimi oktávami šumu.

Z dôvodu homogénnosti a jednotvárnosti sa v dnešnej dobe *fBm* na generovanie terénu príliš nepoužíva. V záujme čo najvyššieho realizmu je nutné použiť algoritmy, ktoré generujú zaujímavejšie heterogénne terény. Cieľom týchto algoritmov je generovať terén, ktorého členitosť je v rôznych miestach odlišná. Fraktály, ktoré generujú takéto terény sa nazývajú *multifraktály*. Ich základný princíp je rovnaký ako pri *fBm* – výsledok sa konštruuje iteratívne zo šumov s rôznymi frekvenciami. Rozdiel je v tom, že podiel každej frekvencie šumu vo výsledku je nejakým spôsobom váhovaný a táto váha je závislá na predchádzajúcich iteráciách. Nižšie sú predstavené dva typy multifraktálov, ktoré generujú zaujímavé terény. Šum používaný v týchto algoritmoch by mal generovať hodnoty v intervale $\langle -1; 1 \rangle$.

[3] https://www.minecraft.net
[4] https://www.nomanssky.com/

2.3.1 Hybridný Multifraktál

Autor tohto algoritmu, Ken Musgrave [3], sa snažil zostrojiť terén, ktorý by implementoval jeho nasledujúce pozorovanie: V skutočnom teréne sú doliny topograficky hladšie a vyššie položené miesta sú kvôli erozívnym procesom členitejšie. Algoritmus mal navyše brať do úvahy, že doliny nevznikajú len v okolí nadmorskej výšky nula metrov, ale môžu vzniknúť aj vyššie. Myslel si, že toto môže dosiahnuť váhovaním vyšších frekvencií šumu vo výsledku na základe lokálnej hodnoty z predchádzajúcej iterácie [3]. Výsledný postup je znázornený v algoritme 2.

Algoritmus 2: Algoritmus na generovanie hybridného multifraktálu [3].

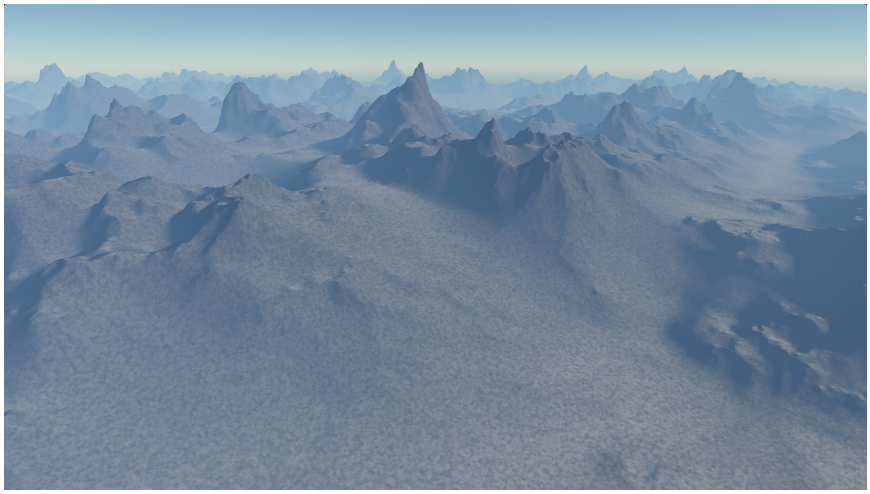
Input: Input point $point$, Hurst exponent H, number of octaves $octaves$, noise offset constant $offset$
Output: Value of hybrid multifractal in a given input point

$result \leftarrow 0$
$f \leftarrow 1$ `// Frequency of the noise`
$a \leftarrow 1$ `// Amplitude of the noise`
$result \leftarrow a \cdot (Noise(point) + offset)$ `// Add the first octave`
$weight \leftarrow result$
for $i = 1$ **to** $(octaves - 1)$ **do**
  if $weight > 1$ **then**
    $weight \leftarrow 1$ `// Prevents divergence`
  end
  $f \leftarrow f \cdot 2$ `// Doubling the frequency of the noise`
  $a \leftarrow f^{-H}$
  $signal \leftarrow a \cdot (Noise(f \cdot point) + offset)$
  `// Weighting based on previous iteration's local value`
  $result \leftarrow result + weight \cdot signal$
  $weight \leftarrow weight \cdot signal$ `// Update the weighting value`
end
return $result$

Tento algortmus obsahuje oproti *fBm* o jeden parameter naviac. Parameter $offset$ slúži na posun hodnôt šumu z intervalu $\langle -1; 1\rangle$ bližšie k intervalu $\langle 0; 2\rangle$. Zaujímavý fakt je, že tento algoritmus nerobí to, o čo sa autor pokúšal. Dolina vo vyššej nadmorskej výške nie je definovaná lokálnou hodnotou predchádzajúcej frekvencie ale lokálnym gradientom funkcie [3]. Napriek tomu tento algoritmus vytvára zaujímavé heterogénne terény, v ktorých sa nachádzajú roviny, menšie hladké kopce a aj vysoké pohoria. Príklad terénu vygenerovaného týmto algoritmom je zobrazený na obrázku 2.6.

2.3.2 Ryhovaný Multifraktál

Ďalší zaujímavý heterogénny model terénu vytvorený Kenom Musgraveom je *ryhovaný multifraktál* (angl. *ridged multifractal*) [3]. Tento fraktál pripomína skalnaté pohoria s ostrými hranami. Algoritmus na generovanie tohto fraktálu je vo výpise 3.



Obr. 2.6: Príklad terénu generovaného pomocou hybridného multifraktálu.

Algoritmus 3: Algoritmus na generovanie ryhovaného multifraktálu [3].

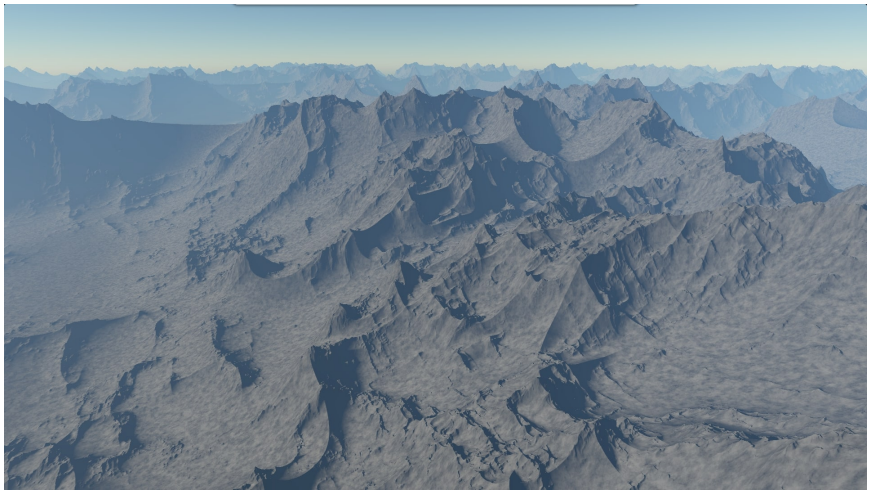
Input: Input point *point*, Hurst exponent H, number of octaves *octaves*, gain constant *gain*

Output: Value of ridged multifractal in a given input point

```
1  result ← 0
2  f ← 1                                          // Frequency of the noise
3  a ← 1                                          // Amplitude of the noise
4  signal ← 1 − |Noise(point)|
5  signal ← signal^2
6  result ← a · signal                            // Add the first octave
7  weight ← 1
8  for i = 1 to (octaves − 1) do
9  |  weight ← signal · gain
10 |  if weight < 0 then
11 |  |  weight ← 0
12 |  end
13 |  else if weight > 1 then
14 |  |  weight ← 1
15 |  end
16 |  f ← f · 2                          // Doubling the frequency of the noise
17 |  a ← f^−H
18 |  signal ← 1 − |Noise(f · point)|
19 |  signal ← signal · weight                  // Weighting the contribution
20 |  result ← result + a · signal
21 end
22 return result
```

Ostré hrany tohoto fraktálu vznikajú použitím absolútnej hodnoty šumu a jeho následným prevrátením $(1 - |Noise|)$. Táto hodnota je navyše umocnená na druhú aby vzniknuté hrany boli ešte ostrejšie. Nový parameter *gain* riadi ako veľmi bude hodnota šumu z predchádzajúcej iterácie ovplyvňovať príspevok aktuálneho šumu k výsledku. Odporúčaná

hodnota tohto parametru je 2. Terén generovaný ako ryhovaný multifraktál je zobrazený na obrázku 2.7.



Obr. 2.7: Príklad terénu modelovaného pomocou ryhovaného multifraktálu.

V tejto kapitole bolo predstavených niekoľko algoritmov na generovanie terénu. Tieto algoritmy vytvoria výškovú mapu, ktorú treba pri vykresľovaní nejakým spôsobom transformovať na 3D geometriu terénu. Jeden zo spôsobov vykresľovania výškovej mapy je popísaný v nasledujúcej kapitole.

Kapitola 3

Realistické zobrazovanie krajiny

Realistické zobrazovanie krajiny má využitie v mnohých oblastiach. Najviac sa využíva v počítačových hrách, filmoch, leteckých simulátoroch, a iných. Najhlavnejšia časť krajiny je pravdepodobne terén. Pri realistickom zobrazovaní krajiny je však veľmi dôležité zachytenie volumetrických javov. Medzi ne patrí atmosféra a oblaky. Z tohto dôvodu je veľká časť tejto kapitoly venovaná volumetrickému vykresľovaniu a šíreniu svetla v objeme, ktoré sú popísané v sekcii 3.1. V sekcii 3.2 je konkrétnejšie popísané vykresľovanie atmosféry a v sekcii 3.3 je popísané vykresľovanie terénu definovaného procedurálnou výškovou mapou.

3.1 Šírenie svetla v médiu

Médium (niekedy nazývané participujúce médium alebo objem) je prostredie, v ktorom sa nachádzajú malé čiastočky (napríklad kvapôčky vody, prach, ...), ktoré menia správanie svetla prechádzajúceho týmto prostredím. Za médium sa považuje napríklad vzduch, voda, para, hmla, atď. Hustota čiastočiek môže byť homogénna (všade rovnaká), ako napríklad vo vode alebo vo vzduchu, alebo heterogénna (hustota je v rôznych miestach rôzna). Príklady médií s heterogénnou hustotou sú para alebo oblaky.

Aby bolo v počítačovej grafike možné popísať správanie svetla pri prechode médiom, je potrebné zaviesť niekoľko zjednodušujúcich predpokladov. V tejto práci sú použité predpoklady, ktoré vo svojej dizertačnej práci [5] zaviedol Wojciech Jarosz:

1. Predpokladá sa, že médium je možné modelovať ako kolekciu mikroskopických čiastočiek.

2. Keďže čiastočky sú mikroskopické a náhodne rozložené, v simulácii sa nereprezentujú všetky čiastočky jednotlivo. Miesto toho sa simulácia zaoberá agregovanými pravdepodobnosťami správania sa svetla počas jeho prechodu médiom.

3. Čiastočky sú od seba veľmi ďaleko (relatívne k ich veľkostiam). Z tohto predpokladu vyplýva, že každá interakcia fotónu s čiastočkou je štatisticky nezávislá od predchádzajúcich interakcií tohto fotónu s čiastočkami.

3.1.1 Interakcia svetla s čiastočkami v médiu

Keď sa fotón pohybuje médiom, ktoré je modelované ako kolekcia čiastočiek, môže buď minúť všetky čiastočky a pohybovať sa ďalej bez zmeny, alebo môže s niektorými čiastočkami

interagovať. Pravdepodobnosť, že interakcia nastane súvisí s *extinkčným koeficientom* σ_t (jednotka $[1/m]$) média. Táto veličina závisí na hustote čiastočiek v médiu a ich veľkosti.

Keď interakcia fotónu nastane, môžu sa stať dve udalosti. Fotón môže byť čiastočkou absorbovaný (zmení sa na inú formu energie, napríklad teplo) alebo môže byť rozptýlený do iného smeru (angl. *outscattering*) [5]. Relatívne pravdepodobnosti týchto dvoch udalostí udáva *koeficient absorpcie* σ_a a *koeficient rozptylu* σ_s. Súčet týchto keoficientov je rovný *extinkčnému koeficientu*: $\sigma_t = \sigma_a + \sigma_s$. Absorpcia a rozptyl spôsobujú zníženie intenzity svetla prechádzajúceho médiom. Na výpočet časti intenzity svetla, ktorá sa zachová po prechode médiom sa používa *Beer-Lambertov zákon*. Tento zákon vyjadruje rovnica (3.1), ktorá slúži na výpočet priehľadnosti média T medzi bodmi $\boldsymbol{x}$ a $\boldsymbol{x'}$. Priehľadnosť rovnako reprezentuje pravdepodobnosť, že fotón prejde médiom medzi danými bodmi bez stretu s čiastočkou, takže jej hodnota je vždy v intervale $\langle 0; 1\rangle$ [11].

Body v 3D priestore (ako napríklad $\boldsymbol{x}$ a $\boldsymbol{x'}$) sú v tomto texte zvýraznené hrubým písmom aby boli lepšie odlíšené od skalárnych hodnôt a smerových vektorov.

$$T(\boldsymbol{x}, \boldsymbol{x'}) = e^{-\tau(\boldsymbol{x}, \boldsymbol{x'})} \tag{3.1}$$

Exponent na pravej strane rovnice (3.1) sa nazýva *optická hĺbka* alebo *optická hrúbka* a označuje sa τ. Táto veličina sa vypočíta integrovaním extinkčných koeficientov média σ_t pozdĺž úsečky medzi bodmi $\boldsymbol{x}$ a $\boldsymbol{x'}$:

$$\tau(\boldsymbol{x}, \boldsymbol{x'}) = \int_0^d \sigma_t(\boldsymbol{x} + t\vec{\omega})\, dt \tag{3.2}$$

kde d je vzdialenosť medzi bodmi $\boldsymbol{x}$ a $\boldsymbol{x'}$ a $\vec{\omega}$ je smerový vektor medzi týmito bodmi. Výraz $\boldsymbol{x} + t\vec{\omega}$, kde $t \in [0, d]$, parametrizuje úsečku medzi bodmi $\boldsymbol{x}$ a $\boldsymbol{x'}$ a $\sigma_t(\boldsymbol{x} + t\vec{\omega})$ reprezentuje extinkčný koeficient v danom bode. Táto rovnica sa používa na výpočet optickej hĺbky heterogénneho média, kde je extinkčný koeficient v rôznych bodoch média rôzny. Pre homogénne médiá s konštantným extinkčným koeficientom je možné výpočet optickej hĺbky zjednodušiť na:

$$\tau(\boldsymbol{x}, \boldsymbol{x'}) = d\, \sigma_t \tag{3.3}$$

kde d je vzdialenosť medzi bodmi $\boldsymbol{x}$ a $\boldsymbol{x'}$ a σ_t je extinkčný koeficient média.

Jedna z užitočných vlastností priehľadnosti, ktorá môže byť použitá pri jej výpočte je, že priehľadnosť medzi troma bodmi na priamke je multiplikatívna:

$$T(\boldsymbol{x}, \boldsymbol{x''}) = T(\boldsymbol{x}, \boldsymbol{x'})\, T(\boldsymbol{x'}, \boldsymbol{x''}) \tag{3.4}$$

pre všetky body $\boldsymbol{x'}$ medzi bodmi $\boldsymbol{x}$ a $\boldsymbol{x''}$ [5].

Pomocou *Beer-Lambertovho zákona* a dosadením rovnice (3.2) alebo (3.3) do rovnice (3.1) je možné vytvoriť rovnicu na výpočet intenzity svetla prechádzajúceho médiom. V tejto rovnici sa berie do úvahy len strata intenzity svetla z dôvodu absorpcie a rozptylu do iného smeru (*outscattering*):

$$L(\boldsymbol{x}, \vec{\omega}) = T(\boldsymbol{x}, \boldsymbol{x'})\, L(\boldsymbol{x'}, \vec{\omega}) \tag{3.5}$$

kde:

$\boldsymbol{x}$ – bod kde svetlo vystupuje z média

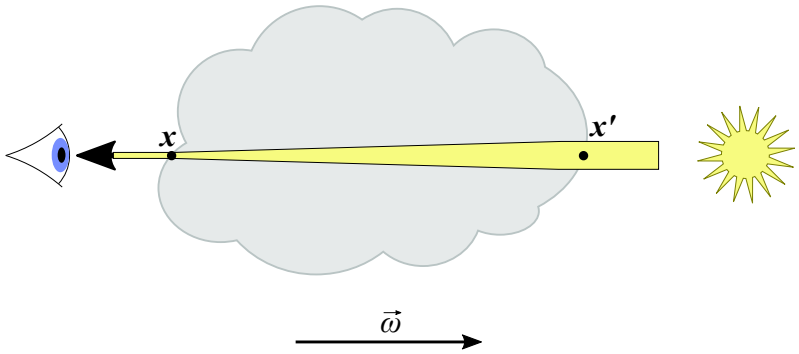
$\boldsymbol{x'}$ – bod kde svetlo vstupuje do média

$\vec{\omega}$ – smer pohľadu pozorovateľa

$T(\boldsymbol{x}, \boldsymbol{x'})$ – priehľadnosť média medzi bodmi $\boldsymbol{x}$ a $\boldsymbol{x'}$ (viď rovnica (3.1))

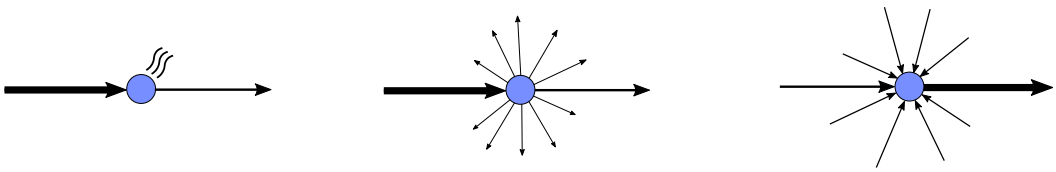
$L(\boldsymbol{x}, \vec{\omega})$ – Intenzita svetla vchádzajúceho do bodu $\boldsymbol{x}$ zo smeru $\vec{\omega}$

Šírenie svetla médiom, ktoré sa počíta rovnicou (3.5), je vizualizované na obrázku 3.1.



Obr. 3.1: Šírenie svetla médiom. Intenzita svetla sa kvôli absorpcii a rozptylu postupne znižuje.

Okrem absorpcie a rozptylu svetla do iného smeru, ktoré znižujú intenzitu svetla prechádzajúceho médiom, môže nastať tretí prípad. Svetlo prichádzajúce z iného smeru sa na čiastočke rozptýli do pozorovaného smeru (angl. *inscattering*) čo vedie k zvýšeniu intenzity svetla v smere pozorovania. Všetky tri typy interakcie svetla s čiastočkami v médiu sú znázornené na obrázku 3.2. Médium ako napríklad oheň, môže navyše ešte svetlo vyžarovať. Toto správanie sa nazýva emisia svetla ale ňou sa táto práca nebude zaoberať.



(a) Absorpcia (b) Rozptyl mimo smer (c) Rozptyl do smeru

Obr. 3.2: Tri typy interakcie svetla s čiastočkou v médiu. Absorpcia a rozptyl svetla mimo pozorovaný smer (*outscaterring*) prispievajú k úbytku intenzity svetla. Rozptyl svetla z okolia do pozorovaného smeru (*inscattering*) prispieva naopak k zvýšeniu intenzity svetla putujúceho pozdĺž pozorovaného smeru.

Svetlo rozptýlené do pozorovaného smeru $Li(\boldsymbol{x}, \vec{\omega})$ reprezentuje intenzitu svetla, ktoré je v bode $\boldsymbol{x}$ rozptýlené do tenkého lúča pozdĺž smerového vektoru $\vec{\omega}$. Svetlo môže vstúpiť do bodu $\boldsymbol{x}$ zo všetkých smerov, takže výpočet rozptýleného svetla zahŕňa integrovanie intenzity prichádzajúceho svetla nad guľovým priestorom smerov $\Omega_{4\pi}$ [5]:

$$Li(\boldsymbol{x}, \vec{\omega}) = \int_{\Omega_{4\pi}} p(\boldsymbol{x}, \vec{\omega}, \vec{\omega}')\, L(\boldsymbol{x}, \vec{\omega}')\, d\vec{\omega}' \tag{3.6}$$

kde $L(\boldsymbol{x}, \vec{\omega}')$ reprezentuje intenzitu svetla prichádzajúceho do bodu $\boldsymbol{x}$ zo smeru $\vec{\omega}'$ a $p(\boldsymbol{x}, \vec{\omega}, \vec{\omega}')$ je fázová funkcia, ktorá je popísaná v nasledujúcej sekcii.

3.1.2 Fázová funkcia

Úloha fázovej funkcie $p(\boldsymbol{x}, \vec{\omega}, \vec{\omega}')$ pri vykresľovaní objemu je podobná ako úloha *BRDF*[1] pri vykresľovaní povrchu. Fázová funkcia popisuje distribúciu svetla rozptýleného v danom bode pre všetky smery prichádzajúceho svetla $\vec{\omega}'$ a všetky smery vychádzajúceho svetla $\vec{\omega}$ [9]. Konkrétna hodnota fázovej funkcie vyjadruje pravdepodobnosť, že svetlo prichádzajúce zo smeru $\vec{\omega}'$ do bodu $\boldsymbol{x}$ je rozptýlené do smeru $\vec{\omega}$ (oba vektory smerujú von z bodu $\boldsymbol{x}$, viď obrázok 3.3). Fázová funkcia je závislá len na kosínuse uhla daných smerových vektorov: $cos\,\theta = \vec{\omega} \cdot \vec{\omega}'$ a platí tu reciprocita: $p(\boldsymbol{x}, \vec{\omega}, \vec{\omega}') = p(\boldsymbol{x}, \vec{\omega}', \vec{\omega})$. Keďže fázová funkcia je distribučná funkcia, jej integrál cez guľovú plochu smerov musí byť pre všetky smerové vektory $\vec{\omega}$ rovný jednej:

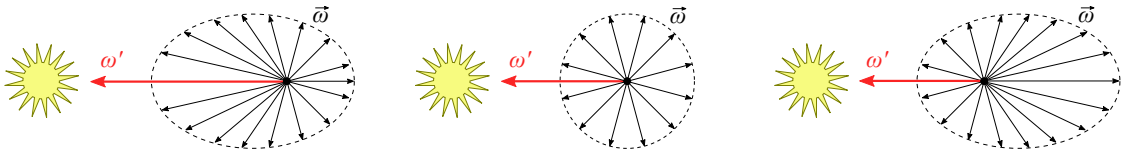
$$\int_{\Omega_{4\pi}} p(\boldsymbol{x}, \vec{\omega}, \vec{\omega}')\, d\vec{\omega}' = 1 \tag{3.7}$$

Priemerný kosínus g uhla medzi smerovými vektormi slúži na určenie dominantného smeru, do ktorého je svetlo v médiu rozptýlené. Jeho hodnoty patria do intervalu $\langle -1, 1 \rangle$ a na jeho výpočet sa používa rovnica (3.8).

$$g = \int_{\Omega_{4\pi}} p(\boldsymbol{x}, \vec{\omega}, \vec{\omega}')\, cos\,\theta\, d\vec{\omega}' \tag{3.8}$$

Podľa dominantného smeru rozptýleného svetla sa rozptyl delí na tri typy (viď obrázok 3.3):

- **Izotropný rozptyl (*Isotropic scattering*)** – Svetlo je v médiu rozpýlené rovnomerne do všetkých smerov. Pri $g = 0$.
- **Predný rozptyl (*Forward scattering*)** – Svetlo je viac rozptýlené dopredu, čiže v smere jeho šírenia. Pri $g > 0$.
- **Zadný rozptyl (*Back scattering*)** – Svetlo je viac rozptýlené dozadu, čiže smerom späť k svetelnému zdroju. Pri $g < 0$.



(a) Zadný rozptyl (b) Izotropný rozptyl (c) Predný rozptyl

Obr. 3.3: Vizualizácia fázovej funkcie $p(\boldsymbol{x}, \vec{\omega}, \vec{\omega}')$. Izotropný rozptyl je najjednoduchší prípad rozptylu, kedy je svetlo rozptýlné rovnomerne do všetkých smerov. Prírodné materiály však najčastejšie rozptyľujú svetlo prednostne buď dopredu, alebo dozadu [5].

Tvar fázovej funkcie záleží na veľkosti a orientácii čiastočiek v médiu. Fázová funkcia je väčšinou rôzna pre rôzne čiastočky v médiu. Pre jednoduchosť sa však používa priemerná fázová funkcia, ktorá zachytáva najdôležitejšie vlastnosti rozptylu v danom médiu [15]. Nižšie je popísaných niekoľko najčastejšie používaných fázových funkcií, ktoré sa používajú pri vykresľovaní média.

[1] *Bidirectional reflectance distribution function* – obojsmerná distribučná funkcia odrazu svetla

Izotropná fázová funkcia

Je to najjednoduchšia fázová funkcia. Svetlo je rozptýlené do náhodného smeru, takže pravdepodobnosť rozptylu je pre všetky smery rovnaká. Izotropná fázová funkcia je definovaná nasledovne:

$$p_I(\boldsymbol{x}, \vec{\omega}, \vec{\omega}') = \frac{1}{4\pi} \tag{3.9}$$

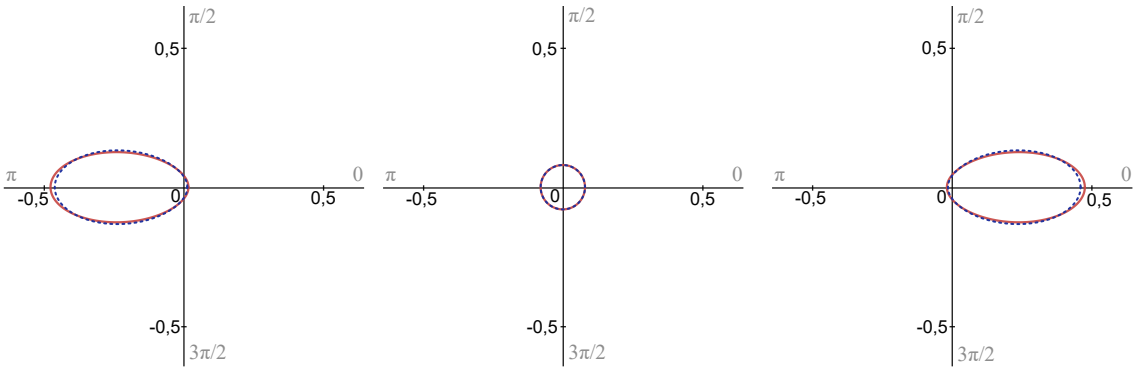
Izotropný rozptyl je volumetrický ekvivalent difúzneho odrazu svetla. Môže sa použiť napríklad pri vykresľovaní homogénnej hmly.

Henyey-Greensteinova fázová funkcia

Je to najviac využívaná neizotropná fázová funkcia. Bola vytvorená v roku 1941 na popis rozptylu svetla v medzigalaktickom prachu [5]. Kvôli jej jednoduchosti sa používa na simuláciu rozptylu svetla v mnohých prírodných materiáloch, ako napríklad vo vode, oblakoch a pokožke. Funkcia závisí len na uhle θ a je definovaná nasledovne:

$$p_{HG}(\theta) = \frac{1}{4\pi} \frac{1 - g^2}{(1 + g^2 - 2g\,cos\,\theta)^{3/2}} \tag{3.10}$$

Parameter g sa nazýva parameter asymetrie a ovplyvňuje dominantný smer rozptylu svetla. Tento parameter reprezentuje priemerný kosínus uhla medzi smerovými vektormi, ktorý bol spomenutý vyššie. Grafy Henyey-Greensteinovej funkcie s rôznymi parametrami g sú zobrazené na obrázku 3.4.



(a) $g = -0,5$ (b) $g = 0,0$ (c) $g = 0,5$

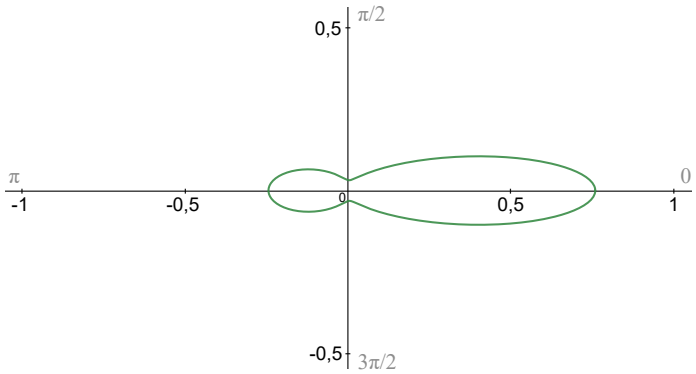
Obr. 3.4: Vizualizácia Henyey-Greensteinovej (červená) a Schlickovej (modrá) fázovej funkcie pre rozdielne hodnoty parametru g. Zobrazené v polárnych súradniciach ako funkcia uhla θ.

Dvojitá Henyey-Greensteinova fázová funkcia

Henyey-Greensteinova fázová funkcia dokáže zachytiť rozptyl len do jedného smeru. Pri niektorých materiáloch je však časť svetla rozptýlená dopredu a časť dozadu. Takýto rozptyl je možné vytvoriť kombináciou dvoch Henyey-Greensteinových funkcií s dvoma rôznymi hodnotami parametru g [15]. Jedna funkcia simuluje predný rozptyl a druhá simuluje zadný rozptyl:

$$p_{HG_2}(\theta) = (1 - f)\, p_{HG}(\theta, g_1) + f\, p_{HG}(\theta, g_2) \tag{3.11}$$

kde $g_1 > 0$ (predný rozptyl) a $g_2 < 0$ (zadný rozptyl). Parameter f slúži ako váha pri interpolácii medzi dvoma funkciami. Graf tejto funkcie je vykreslený na obrázku 3.5.



Obr. 3.5: Graf dvojitej Henyey-Greensteinovej fázovej funkcie s parametrami $g_1 = 0,7$; $g_2 = -0,5$; $f = 0,5$. Zobrazené v polárnych súradniciach ako funkcia uhla θ.

Schlickova fázová funkcia

Henyey-Greensteinova fázová funkcia je intuitívna a flexibilná, avšak výpočet zlomkového exponentu v menovateli môže byť niekedy časovo náročný. V takýchto prípadoch je možné túto fázovú funkciu nahradiť Schlickovou fázovou funkciou. Táto fázová funkcia dobre aproximuje tvar Henyey-Greensteinovej fázovej funkcie pomocou elipsoidu a je efektívnejšia na výpočet [5]. Schlickova fázová funkcia je definovaná nasledovne:

$$p_S(\theta) = \frac{1}{4\pi} \frac{1 - k^2}{(1 - k\ cos\,\theta)^2} \tag{3.12}$$

kde $k \in (-1, 1)$, podobne ako parameter g, ovplyvňuje dominantný smer rozptylu. Rovnako ako pri parametri g, $k = 0$ spôsobuje izotropný rozptyl, záporné hodnoty spôsobujú zadný a kladné hodnoty predný rozptyl. Bolo zistené [5], že Schlickova fázová funkcia s parametrom $k = 1,55g - 0,55g^3$ najlepšie aproximuje Henyey-Greensteinovu fázovú funkciu s parametrom g. Porovnanie grafov oboch funkcií je na obrázku 3.4.

3.1.3 Rovnica vykresľovania objemu

Pomocou teórie o šírení svetla v médiu vysvetlenej v minulých sekciách je možné vytvoriť kompletný model šírenia svetla v médiu a takisto aj zostrojiť rovnicu na vykresľovanie objemu:

$$L(\boldsymbol{x}, \vec{\omega}) = \underbrace{T(\boldsymbol{x}, \boldsymbol{x}_d)\, L(\boldsymbol{x}_d, -\vec{\omega})}_{\text{svetlo odrazené od povrchu}} + \underbrace{\int_0^d T(\boldsymbol{x}, \boldsymbol{x}_t)\, \sigma_s(\boldsymbol{x}_t)\, L_i(\boldsymbol{x}_t, -\vec{\omega})\, dt}_{\text{svetlo rozptýlené do pozorovaného smeru}} \tag{3.13}$$

kde:

$L(\boldsymbol{x}, \vec{\omega})$ – intenzita svetla vchádzajúceho do bodu $\boldsymbol{x}$ zo smeru $\vec{\omega}$

$\boldsymbol{x}_d = \boldsymbol{x} + d\vec{\omega}$ – bod nachádzajúci sa na povrchu za médiom (viď obrázok 3.6)

d – hrúbka média z bodu $\boldsymbol{x}$ v smere $\vec{\omega}$

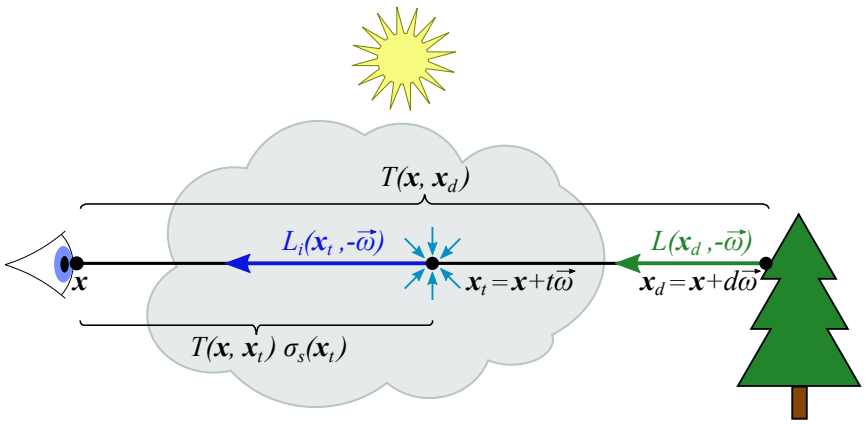
$\boldsymbol{x}_t = \boldsymbol{x} + t\vec{\omega}$ – kde $t \in (0, d)$

$\sigma_s(\boldsymbol{x}_t)$ – koeficient rozptylu v bode $\boldsymbol{x}_t$

T – priehľadnosť média medzi danými bodmi (viď rovnica (3.1))

$L_i(\boldsymbol{x}_t, -\vec{\omega})$ – svetlo rozptýlené do smeru $-\vec{\omega}$ v bode $\boldsymbol{x}_t$ (viď rovnica (3.6))

Rovnica (3.13) sa skladá z dvoch výrazov. Prvý výraz vyjadruje intenzitu svetla, ktoré bolo odrazené od povrchu za médiom a prešlo médiom až k pozorovateľovi. Druhý výraz vyjadruje intenzitu svetla, ktoré bolo v médiu rozptýlené z okolia do pozorovaného smeru. Vizualizácia rovnice (3.13) je na obrázku 3.6.



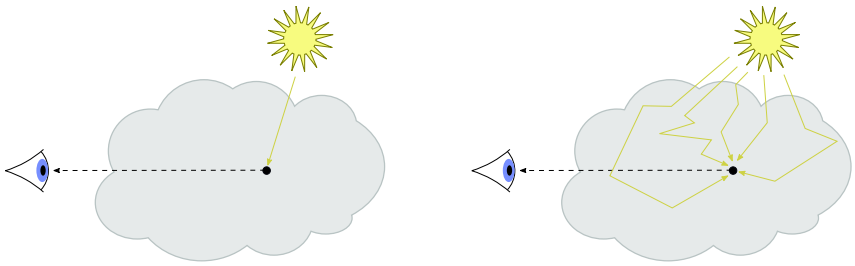
Obr. 3.6: Intenzita svetla prichádzajúceho k pozorovateľovi $L(\boldsymbol{x}, \vec{\omega})$ sa skladá zo zníženej (v dôsledku absorpcie a rozptylu) intenzity svetla odrazeného z najbližšieho viditeľného povrchu $L(\boldsymbol{x}_d, -\vec{\omega})$ a z akumulovaného svetla rozptýleného do pozorovaného smeru $L_i(\boldsymbol{x}_t, -\vec{\omega})$. Inšpirácia na obrázok z [5].

Rovnica (3.13) sa používa v počítačovej grafike na vykresľovanie scén s prítomnosťou participujúceho média. Táto rovnica je však veľmi zložitá na výpočet. Intenzita svetla v každom bode média je závislá na intenzite svetla v každom inom bode média a na intenzite svetla odrazeného zo všetkých okolitých povrchov. Táto rovnica sa môže riešiť rôznymi spôsobmi, medzi najznámejšie a najpoužívanejšie patria *path tracing*, *photon mapping* a *ray-marching*. V tejto práci je bližšie popísaná posledná spomenutá technika.

3.1.4 Riešenie rovnice vykresľovania objemu v reálnom čase

Cieľ tejto práce je vykresľovanie v reálnom čase, takže rovnica vykresľovania objemu, predstavená v sekcii 3.1.3, musí byť značne zjednodušená, rovnako ako aj jej riešenie. Výpočet intenzity svetla rozptýleného do pozorovaného smeru L_i, ktoré sa počíta rovnicou 3.6, zahŕňa integrovanie intenzity svetla prichádzajúceho zo všetkých okolitých smerov. Táto intenzita sa počíta rovnako rovnicou 3.6. Je to rekurzívny výpočet pričom každý ďalší level rekurzie počíta rozptyl vyššieho rádu. Toto je dôvod prečo je riešenie rovnice vykresľovania objemu tak zložité. Zjednodušenie, ktoré umožní vykresľovanie v reálnom čase je, že sa nebude

brať do úvahy viacnásobný rozptyl ale iba jednonásobný. Jednonásobný rozptyl znamená, že svetlo putujúce zo svetelného zdroja k pozorovateľovi sa na čiastočke v médiu rozptýli iba raz (viď obrázok 3.7). Takýto prípad keď je svetlo rozptýlené iba raz sa pri vykresľovaní objemu nazýva priame osvetlenie a svetlo rozptýlené viackrát sa nazýva nepriame osvetlenie.



Obr. 3.7: Porovnanie priameho (vľavo) a nepriameho (vpravo) osvetlenia v médiu. Priame osvetlenie je svetlo prichádzajúce priamo zo svetelného zdroja do daného bodu. Nepriame osvetlenie integruje svetlo prichádzajúceho do daného bodu zo všetkých smerov (rovnica (3.6)).

Na vyriešenie takto zjednodušenej rovnice vykresľovania objemu je možné použiť *raymarching*. Ide o techniku podobnú *raytracingu*, pri ktorej sa z kamery vyšle primárny lúč cez médium až po prvý viditeľný povrch alebo koniec média. Po tomto lúči sa postupuje po malých krokoch, pričom v každom kroku sa vzorkuje hustota média, počíta sa priame osvetlenie a integruje priehľadnosť média [1]. Priame osvetlenie je počítané *raymarchingom* sekundárneho lúča smerujúceho k svetelnému zdroju. Tento postup je vizualizovaný na obrázku 3.8.

Diskretizovaná rovnica vykresľovania objemu prispôsobená pre *raymarching* je definovaná nasledovne:

$$L(\boldsymbol{x},\vec{\omega}) = T(\boldsymbol{x},\boldsymbol{x}_d)\,L(\boldsymbol{x}_d,-\vec{\omega}) + \sum_{n=0}^{N-1} T(\boldsymbol{x},\boldsymbol{x}_n)\,\sigma_s(\boldsymbol{x}_n)\,p(\boldsymbol{x}_n,-\vec{\omega},\vec{\omega_s})\,L(\boldsymbol{x}_n,\vec{\omega_s})\,\Delta_x \tag{3.14}$$

kde:

N – počet vzoriek pozdĺž primárneho lúča

$\Delta_x = \frac{d}{N}$ – veľkosť kroku *raymarchingu*

d – hrúbka média z bodu $\boldsymbol{x}_t$ v smere $\vec{\omega}$

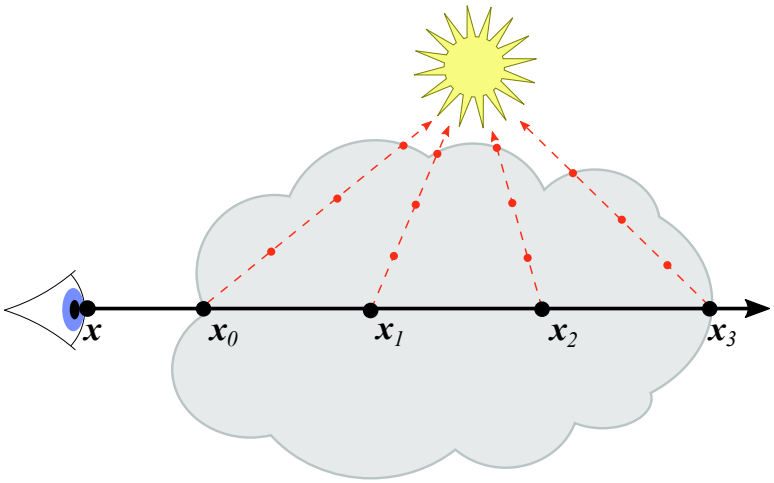
$\boldsymbol{x}_n = \boldsymbol{x} + n\,\Delta_x$ – pozícia aktuálne vzorkovaného bodu

$\vec{\omega_s}$ – smerový vektor smerujúci z bodu $\boldsymbol{x}_n$ k svetelnému zdroju

$L(\boldsymbol{x}_n,\vec{\omega_s})$ – intenzita svetla prichádzajúceho do bodu $\boldsymbol{x}_n$ priamo zo svetelného zdroja

$\boldsymbol{x}_d, \sigma_s, T$ – rovnaký význam ako v rovnici (3.13)

Zložka $L(\boldsymbol{x}_n,\vec{\omega_s})$ v rovnici (3.14) vyjadruje priame osvetlenie počítané *raymarchingom* sekundárneho lúča. Táto zložka nahrádza zložku $L_i(\boldsymbol{x}_t,-\vec{\omega})$ z rovnice (3.13).



Obr. 3.8: Znázornenie *raymarchingu* média. Primárny lúč (čierny) smeruje od pozorovateľa cez médium. Sekundárne lúče (červené) smerujú z každého vzorkovaného bodu ($\boldsymbol{x}_0$ – $\boldsymbol{x}_3$) k svetelnému zdroju.

3.2 Simulácia atmosféry

Realistické vykresľovanie atmosféry je zložitý problém, ktorého riešenie je však pri zobrazovaní krajiny veľmi dôležité. Realistická atmosféra umožňuje vytvoriť oblohu, ktorá sa interaktívne mení v závislosti na polohe Slnka.

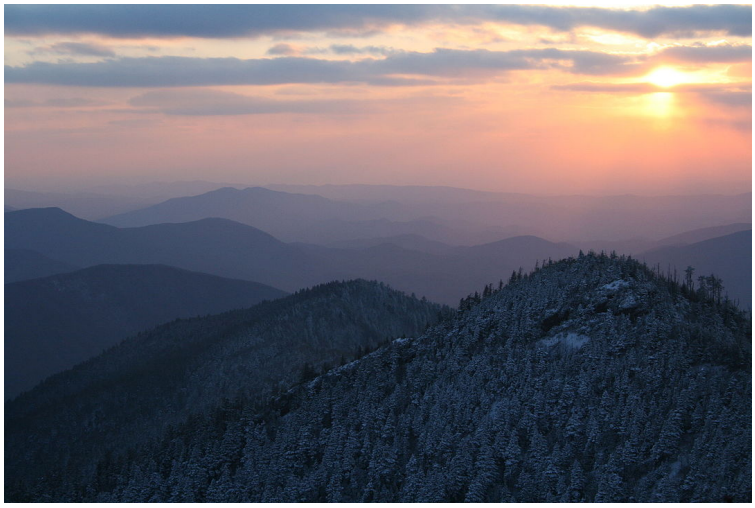
Atmosféra je médium skladajúce sa z rôznych čiastočiek, takže jej vykresľovanie je založené na poznatkoch z kapitoly 3.1. Cieľom tejto sekcie je popísať ako rozptyl svetla na čiastočkách v atmosfére spôsobuje oranžové sfarbenie oblohy počas východu a západu Slnka, modré sfarbenie cez deň, a ako je možné tento rozptyl simulovať v počítačovej grafike.

Rozptyl svetla v atmosfére nespôsobuje len sfarbenie oblohy, ale ovplyvňuje aj vzhľad vzdialených objektov v krajine. So zväčšujúcou sa vzdialenosťou medzi pozorovateľom a objektom sú farby objektu postupne nahradené farbou atmosféry (modrou cez deň a oranžovo-červenou pri východe a západe Slnka, viď obrázok 3.9) [21]. Tento efekt sa nazýva vzdušná perspektíva (anglicky *aerial perspective*). Pre ľudí je to dôležitá vizuálna pomôcka na rozpoznanie vzdialenosti v krajine a preto je pri vykresľovaní krajiny v počítačovej grafike tento efekt dôležitý.

Charakteristika rozptylu svetla v atmosfére záleží na veľkosti čiastočiek, ktoré sa v nej nachádzajú. Rozptyl svetla na malých čiastočkách, ako sú vzduchové molekuly, sa nazýva Rayleighov rozptyl a rozptyl na veľkých čiastočkách (aerosóloch), ako napríklad prach, sa nazýva Mieho rozptyl [13]. Tieto dva typy rozptylu sú bližšie popísané v sekcii 3.2.2 a v sekcii 3.2.3.

3.2.1 Model atmosféry

Model atmosféry popísaný v tejto práci je založený na modeli vytvorenom Tomoyuki Nishitom v roku 1993 [13], ktorý sa v grafických aplikáciách často používa dodnes. Atmosféra je modelovaná ako guľa okolo planéty, ktorá má nad povrchom terénu určitú hrúbku. Hrúbka atmosféry Zeme je približne 100 km [21]. V atmosfére sa nachádzajú čiastočky, konkrétne vzduchové molekuly a aerosóly. Dôležitá vlastnosť atmosféry je, že hustota čiastočiek



Obr. 3.9: Fotografia znázorňujúca vzdušnú perspektívu počas západu Slnka. Farba pohoria blízko pri pozorovateľovi nie je nijako ovplyvnená farbou atmosféry. So zväčšujúcou sa vzdialenosťou sa však v dôsledku rozptylu svetla v atmosfére mení farba vzdialených pohorí na oranžovo-červenú.[2]

sa s výškou znižuje a hustota vplýva na množstvo rozptýleného svetla – čím menšia hustota čiastočiek, tým menšia pravdepodobnosť rozptylu. V použitom modeli sa predpokladá, že hustota čiastočiek sa s výškou znižuje exponenciálne. Hustota čiastočiek v atmosfére ρ v nadmorskej výške h sa počíta nasledovne:

$$\rho(h) = \rho(0)\, e^{-\frac{h}{H_0}} \tag{3.15}$$

kde H_0 je anglicky nazývané *scale height* a táto veličina predstavuje hrúbku atmosféry keby bola hustota čiastočiek všade rovnaká [13]. Typicky používané hodnoty sú $H_0 = 8\,km$ pre vzduchové molekuly spôsobujúce Rayleighov rozptyl a $H_0 = 1,2\,km$ pre aerosóly spôsobujúce Mieho rozptyl [4].

Keďže atmosféra je participujúce médium, slnečné svetlo sa pri prechode atmosférou šíri rovnako ako bolo popísané v kapitole 3.1 – svetlo je čiastočkami buď absorbované, alebo rozptýlené. Viacnásobný rozptyl je ignorovaný, pretože jeho príspevok je zanedbateľný a výpočet je náročný. Príspevok svetla odrazeného od povrchu planéty do atmosféry môže ovplyvniť výslednú farbu oblohy [21], ale kvôli zložitosti výpočtu je tiež ignorovaný. V tomto modeli sa predpokladá, že svetlo sa šíri po priamke aj keď v skutočnosti sa kvôli rozdielnym indexom lomu v rôznych nadmorských výškach šíri po krivke [13].

Hlavný zdroj svetla v atmosfére je Slnko. Slnko je od Zeme tak ďaleko, že sa môže predpokladať, že slnečné lúče prichádzajúce do atmosféry sú na seba rovnobežné [21]. Tento predpoklad značne zjednoduší výpočet farby atmosféry popísaný nižšie.

3.2.2 Rayleighov rozptyl

Rayleighov rozptyl je spôsobený malými vzduchovými molekulami. Tento rozptyl je silne závislý na vlnovej dĺžke svetla. Presnejšie, vzduchové molekuly rozptyľujú viac svetlo s kratšou vlnovou dĺžkou – modré svetlo je teda rozptýlené viac ako zelené a červené [21]. Preto je

[2]Fotografia prevzatá z https://en.wikipedia.org/wiki/Aerial_perspective

pri vykresľovaní atmosféry potrebné počítať rozptyl pre každú farebnú zložku svetla zvlášť. V tejto práci sú použité vlnové dĺžky $440\,nm$, $550\,nm$ a $680\,nm$, ktoré odpovedajú modrej, zelenej a červenej farbe svetla. Koeficient rozptylu Rayleighovho rozptyl σ_s^R v nadmorskej výške h, pre svetlo s vlnovou dĺžkou λ sa vypočíta podľa nasledujúcej rovnice:

$$\sigma_s^R(h,\lambda) = \frac{8\pi^3\,(n^2-1)^2}{3N\lambda^4}\,e^{-\frac{h}{H_R}} \tag{3.16}$$

kde n je index lomu vzduchu, N je molekulárna hustota vo výške $0\,m$ a H_R je *scale height* pre Rayleighov rozptyl. Ako bolo spomenuté vyššie, typická hodnota je $H_R = 8\,km$.

Pri dosadení kratších vlnových dĺžok (modré svetlo) do rovnice (3.16) je výsledok väčší ako pri dosadení dlhších vlnových dĺžok (červené svetlo). Toto vysvetľuje prečo je obloha cez deň modrá. Pri šírení svetla atmosférou je modré svetlo viac rozptýlené smerom k pozorovateľovi ako zelené a červené svetlo [21]. Pri západe a východe Slnka, keď sa Slnko nachádza nad horizontom, musí slnečné svetlo prejsť v atmosfére dlhšiu vzdialenosť kým sa dostane k pozorovateľovi. Počas tejto vzdialenosti sa väčšina modrého a zeleného svetla rozptýli preč, a ostane len veľká časť červeného svetla, ktoré je rozptyľované najmenej.

Keďže prvý člen rovnice (3.16) je konštantný, na zjednodušenie výpočtu je vhodné zameniť ho za predpočítané hodnoty uvedené v tabuľke 3.1.

Vlnová dĺžka λ $[nm]$	Koeficient rozptylu σ_s^R $[m^{-1}]$
440	$33{,}1 \cdot 10^{-6}$
550	$13{,}5 \cdot 10^{-6}$
680	$5{,}8 \cdot 10^{-6}$

Tabuľka 3.1: Hodnoty koeficientov rozptylu pre Rayleighov rozptyl vo výške $0\,m$ [4].

Pri malých vzduchových molekulách, ktoré spôsobujú Rayleighov rozptyl, je absorpcia svetla zanedbateľná, takže pre extinkčný koeficient platí: $\sigma_t^R = \sigma_s^R$ [4]. Ďalšia dôležitá charakteristika média je fázová funkcia. Rayleighova fázová funkcia je v rovnici (3.17) [21].

$$P_R(\theta) = \frac{3}{16\pi}\,(1 + cos^2(\theta)) \tag{3.17}$$

Uhol θ je jediný parameter Rayleighovej fázovej funkcie. Je to uhol medzi smerovým vektorom k svetelnému zdroju a smerovým vektorom k pozorovateľovi, popísaný v sekcii 3.1.2.

3.2.3 Mieho rozptyl

Mieho rozptyl je spôsobený veľkými čiastočkami v atmosfére nazvanými aerosóly (napr. kvapôčky vody, prach alebo iné nečistoty). Tieto čiastočky rozptyľujú všetky vlnové dĺžky svetla približne rovnako [14]. Mieho rozptyl spôsobuje veľké biele halo okolo Slnka alebo šedú oblohu keď je hmlisto. Na výpočet koeficientu rozptylu pre Mieho rozptyl σ_s^M je možné použiť hodnotu koeficientu rozptylu v nadmorskej výške nula:

$$\sigma_s^M(h,\lambda) = \sigma_s^M(0,\lambda)\,e^{-\frac{h}{H_M}} \tag{3.18}$$

kde h je nadmorská výška, λ je vlnová dĺžka svetla a H_M je *scale height* pre Mieho rozptyl, ktorej hodnota je $1{,}2\,km$. Za hodnotu koeficientu rozptylu pre Mieho rozptyl vo výške $0\,m$

sa zvyčajne volí hodnota: $\sigma_s^M \geq 2 \cdot 10^{-6}\, m^{-1}$ [4]. Hodnota by mala byť približne rovnaká pre všetky vlnové dĺžky.

Aerosóly na rozdiel od vzduchových molekúl časť svetla absorbujú, preto sa za extinkčný koeficient volí nasledujúca hodnota: $\sigma_t^M = 1,11 \cdot \sigma_s^M$. Mieho fázová funkcia je definovaná nasledovne:

$$P_M(\theta) = \frac{3}{8\pi} \frac{(1-g^2)\,(1+cos^2(\theta))}{(2+g^2)\,(1+g^2-2g\,cos(\theta))^{\frac{3}{2}}} \tag{3.19}$$

kde parameter g, rovnako ako pri fázových funkciách v sekcii 3.1.2, ovplyvňuje dominantný smer rozptylu svetla. Aerosóly rozptyľujú svetlo prednostne dopredu, preto by hodnota g mala byť kladná. Pre Mieho rozptyl v atmosfére sa často používa hodnota $g = 0,76$ [21].

3.2.4 Vykresľovanie farby oblohy

Keďže atmosféra je participujúce médium, pri jej vykresľovaní sa používajú rovnaké princípy ako pri vykresľovaní média, ktoré boli popísané v sekcii 3.1.3. Na vykreslenie farby oblohy je možné použiť mierne upravenú rovnicu vykresľovania objemu (rovnica (3.13)). Z tejto rovnice sa odstráni prvý člen, ktorý reprezentuje svetlo odrazené od povrchu nachádzajúceho sa za vykresľovaným médiom, pretože v tejto práci sa predpokladá že mimo atmosféru sa nenachádzajú žiadne vykresľované objekty. Z rovnice (3.13) ostane len člen reprezentujúci svetlo rozptýlené do pozorovaného smeru – v prípade modelu atmosféry použitého v tejto práci to je len priame svetlo zo Slnka rozptýlené do pozorovaného smeru:

$$L(\boldsymbol{x}, \vec{\omega}) = \int_0^d T(\boldsymbol{x}, \boldsymbol{x}_t)\, L_S(\boldsymbol{x}_t, -\vec{\omega})\, dt \tag{3.20}$$

kde:

$L(\boldsymbol{x}, \vec{\omega})$ – farba oblohy pozorovaná z bodu $\boldsymbol{x}$ v smere $\vec{\omega}$

d – vzdialenosť bodu $\boldsymbol{x}$ k priesečníku s okrajom atmosféry $\boldsymbol{x}_d$ v smere $\vec{\omega}$

$\boldsymbol{x}_t = \boldsymbol{x} + t\vec{\omega}$ – kde $t \in (0, d)$

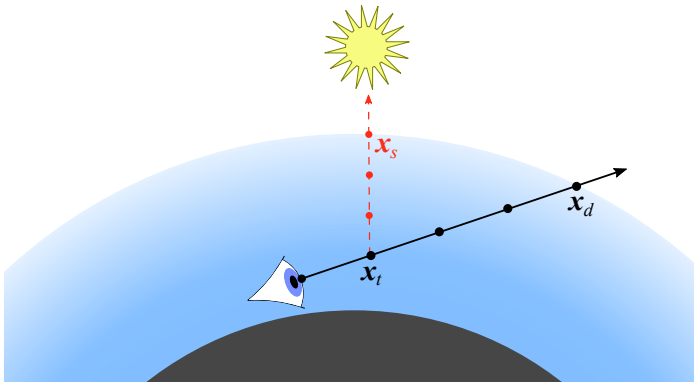
T – priehľadnosť média medzi danými bodmi (viď rovnica (3.1))

$L_S(\boldsymbol{x}_t, -\vec{\omega})$ – intenzita slnečného svetla rozptýleného v bode $\boldsymbol{x}_t$ do smeru $-\vec{\omega}$

Rovnica (3.20) vyjadruje, že na výpočet farby atmosféry je potrebné integrovať priame osvetlenie zo Slnka pozdĺž lúča smerujúceho od pozorovateľa k s okrajom atmosféry (viď obrázok 3.10). Intenzita slnečného svetla L_S rozptýleného v bode $\boldsymbol{x}_t$ do smeru $-\vec{\omega}$ je vynásobená priehľadnosťou atmosféry T medzi bodmi $\boldsymbol{x}$ a $\boldsymbol{x}_t$. Týmto sa získa časť intenzity L_S, ktorá doputuje až do bodu $\boldsymbol{x}$, bez toho aby bola atmosférou absorbovaná alebo rozptýlená. Priame osvetlenie zo Slnka sa vypočíta nasledovne:

$$L_S(\boldsymbol{x}, \vec{\omega}) = \sigma_s(\boldsymbol{x})\, p(\boldsymbol{x}, \vec{\omega}, \vec{\omega_S})\, T(\boldsymbol{x}, \boldsymbol{x}_S)\, I_S \tag{3.21}$$

kde $\sigma_s(\boldsymbol{x})$ je koeficient rozptylu v bode $\boldsymbol{x}$, p je fázová funkcia, $\vec{\omega_S}$ je smerový vektor k Slnku, $\boldsymbol{x}_S$ je priesečník s atmosférou v smere $\vec{\omega_S}$ a I_S je intenzita slnečného svetla (konštanta definovaná používateľom). Rovnice (3.20) a (3.21) je možné vyriešiť pomocou *raymarchingu*, čo je znázornené na obrázku 3.10. Keďže v atmosfére nastáva Rayleighov a aj Mieho rozptyl, tieto rovnice je potrebné vyriešiť dvakrát s dosadením koeficientov a fázových funkcií pre konkrétny rozptyl. Výsledná farba oblohy je súčet výsledkov oboch rozptylov.



Obr. 3.10: Znázornenie *raymarchingu* na vykresľovanie farby atmosféry. Okraj atmosféry je reprezentovaný guľou okolo povrchu planéty, takže potrebné priesečníky ($\boldsymbol{x}_d$, $\boldsymbol{x}_S$) sa nájdu algoritmom na výpočet priesečníka polpriamky a gule.

Rovnicu (3.20) je možné o niečo zjednodušiť [21], čo bude použité pri návrhu algoritmu na vykresľovanie oblohy v kapitole 4.3. Dosadením rovnice (3.21) do rovnice (3.20) sa získa rovnica (3.22). Výsledok fázovej funkcie je konštanta, rovnako ako aj intenzita slnečného svetla I_S. Tieto konštanty sa môžu presunúť pred integrál čím vznikne rovnica (3.23). Pri dosadení rovnice (3.1) do rovnice (3.23) sa môže využiť nasledujúci fakt o mocninách: $e^a \cdot e^b = e^{a+b}$. Týmto vznikne výsledná rovnica (3.24).

$$L(\boldsymbol{x}, \vec{\omega}) = \int_0^d T(\boldsymbol{x}, \boldsymbol{x}_t)\, \sigma_s(\boldsymbol{x}_t)\, p(\boldsymbol{x}_t, -\vec{\omega}, \vec{\omega_S})\, T(\boldsymbol{x}_t, \boldsymbol{x}_S)\, I_S\, dt \tag{3.22}$$

$$L(\boldsymbol{x}, \vec{\omega}) = p(\boldsymbol{x}_t, -\vec{\omega}, \vec{\omega_S})\, I_S \int_0^d T(\boldsymbol{x}, \boldsymbol{x}_t)\, T(\boldsymbol{x}_t, \boldsymbol{x}_S)\, \sigma_s(\boldsymbol{x}_t)\, dt \tag{3.23}$$

$$L(\boldsymbol{x}, \vec{\omega}) = p(\boldsymbol{x}_t, -\vec{\omega}, \vec{\omega_S})\, I_S \int_0^d e^{-\tau(\boldsymbol{x}, \boldsymbol{x}_t) - \tau(\boldsymbol{x}_t, \boldsymbol{x}_S)}\, \sigma_s(\boldsymbol{x}_t)\, dt \tag{3.24}$$

3.2.5 Vykresľovanie vzdušnej perspektívy

Vykresľovanie vzdušnej perspektívy je podobné ako vykresľovanie farby oblohy. Keďže vzdušná perspektíva sa aplikuje na objekty v atmosfére, do jej výpočtu sa navyše pridá svetlo odrazené od objektu za atmosférou (farba objektu). Na rozdiel od výpočtu farby oblohy sa nepočíta rozptyl svetla medzi pozorovateľom a okrajom atmosféry, ale medzi pozorovateľom a objektom na ktorý je vzdušná perspektíva aplikovaná. Rovnica na jej vykresľovanie má už oba členy z rovnice vykresľovania objemu:

$$L(\boldsymbol{x}, \vec{\omega}) = T(\boldsymbol{x}, \boldsymbol{x}_d)\, L(\boldsymbol{x}_d, -\vec{\omega}) + \int_0^d T(\boldsymbol{x}, \boldsymbol{x}_t)\, L_S(\boldsymbol{x}_t, -\vec{\omega})\, dt \tag{3.25}$$

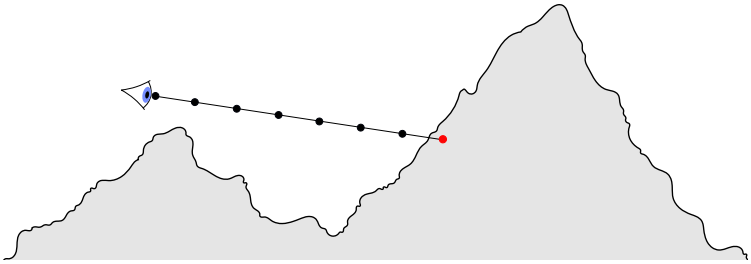
kde d je vzdialenosť medzi pozorovateľom (bodom $\boldsymbol{x}$) a prvým viditeľným bodom na povrchu (bodom $\boldsymbol{x}_d$). Ostatné veličiny majú rovnaký význam ako v rovnici (3.20).

3.3 Vykresľovanie procedurálnej výškovej mapy terénu

V kapitole 2.3 bolo popísané ako vytvoriť výškovú mapu terénu a v tejto kapitole je popísané ako danú výškovú mapu vykresliť. Táto výšková mapa nebude braná ako 2D textúra, ale ako

implicitná funkcia, ktorej vstupom je 2D súradnica v priestore a jej výstupom je výška terénu v danom mieste.

Cieľom vykresľovania výškovej mapy je pre každý pixel obrazu nájsť či existuje priesečník s terénom a prípadne kde sa nachádza. Toto je možné dosiahnuť *raymarchingom*. Rovnako ako pri vykresľovaní objemu sa z kamery vyšle lúč do scény. Po tomto lúči sa postupuje po malých krokoch a v každom kroku sa vyhodnocuje implicitná funkcia terénu. Postupuje sa kým nie je aktuálny vzorkovaný bod nižšie ako aktuálna výška terénu alebo sa nedosiahol maximálny počet krokov. Za priesečník sa môže zvoliť buď prvý bod nižší ako terén, alebo jeho predchodca. Na zvýšenie presnosti je možné vypočítať priesečník interpoláciou aktuálneho a predchádzajúceho vzorkovaného bodu [16].

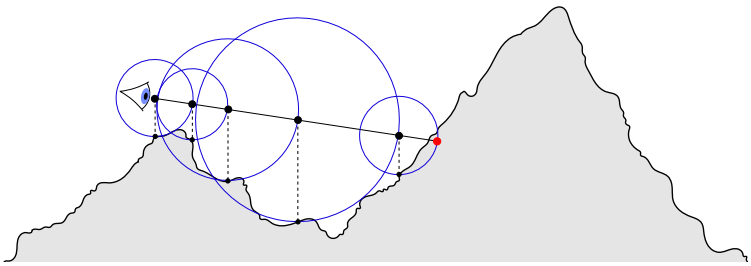


Obr. 3.11: Vykresľovanie terénu pomocou *raymarchingu*. Po lúči sa postupuje po malých krokoch, až kým sa vzorkovaný bod nenachádza pod terénom, čo značí nájdenie priesečníka.

Presnosť a rýchlosť tejto techniky záleží na veľkosti krokov. Kratšie kroky zabezpečia presné nájdenie priesečníkov ale za cenu dlhšieho času vykresľovania. Napriek voľbe veľkosti kroku je táto technika nevhodná na vykresľovanie v reálnom čase. Jej ilustrácia je zobrazená na obrázku 3.11.

Na urýchlenie tejto techniky sa je možné inšpirovať technikou nazvanou *sphere tracing*. Táto technika je založená na rovnakom princípe ako *raymarching*. Po lúči sa postupuje po krokoch, ale veľkosť každého kroku je odhadnutá tzv. znamienkovou funkciou vzdialenosti (angl. *signed distance function – SDF*) [7]. Táto funkcia počíta minimálnu znamienkovú vzdialenosť od daného bodu k implicitnej ploche (*SDF* môže vrátiť menšiu vzdialenosť ako je aktuálna vzdialenosť, ale nikdy nie väčšiu). Znamienková vzdialenosť znamená, že pre body nachádzajúce sa na vnútornej strane implicitnej plochy je táto vzdialenosť záporná a pre body na vonkajšej strane implicitnej plochy je znamienková vzdialenosť kladná. Ak sa za dĺžku kroku *sphere tracingu* zvolí vzdialenosť odhadnutá *SDF*, je isté že krok neprejde cez implicitnú plochu. Pre terén definovaný výškovou mapou sa nedá jednoducho vytvoriť funkcia na odhad vzdialenosti. Za tento odhad sa však dá zobrať výška terénu v danom bode. Táto vzdialenosť vo väčšine prípadov nie je najkratšia vzdialenosť k terénu ale dokáže poslúžiť ako dostatočne dobrý odhad veľkosti kroku, ktorého použitím sa rýchlosť vykresľovania oproti konštantnému kroku výrazne zvýši, aj keď za cenu zníženej presnosti. Presnosť je možné zvýšiť skrátením každého odhadnutého kroku napríklad nasledovne: $krok = a \cdot odhad$, pre $a \in (0, 1)$. Algoritmus sa vykonáva až kým odhadnutá vzdialenosť k terénu pre aktuálny vzorkovaný bod nie je menšia ako predom určená prípustná chyba ε (*epsilon*), kým sa nedosiahne maximálna vykresľovacia vzdialenosť alebo kým sa nevykoná maximálny počet krokov. Ilustrácia tohto algoritmu je na obrázku 3.12.

Ďalšia optimalizácia sa dá dosiahnuť využitím znalosti, že čím ďalej je objekt od kamery, tým menšia je jeho veľkosť na obrazovke a tým menšie sú jeho detaily. Úroveň detailov sa v skutočnosti znižuje lineárne so zväčšujúcou sa vzdialenosťou od kamery [16], takže zvolená



Obr. 3.12: Technika *sphere tracing* upravená pre vykresľovanie terénu. Na odhad dĺžky kroku je použitá vzdialenosť aktuálne vzorkovaného bodu od terénu v danom bode.

chyba vykresľovania ε môže rásť lineárne. Táto optimalizácia urýchli vykresľovanie a pomôže predísť aliasingu. Vyššie popísaný postup je naznačený v algoritme 4.

Algoritmus 4: Vykresľovanie výškovej mapy pomocou *raymarchingu*.

Input: Ray origin ro and direction rd, height map function $terrainMap()$, maximal number of steps $maxSteps$, $nearPlane$, $farPlane$, step size decrease constant a, allowed error $epsilon$

Output: Intersection distance if intersection was found or -1 if there is no intersection

$totalDist \leftarrow nearPlane$
$epsilon \leftarrow epsilon \cdot 0,001$
$steps \leftarrow 0$
while $steps < maxSteps$ **do**
    $samplePoint \leftarrow ro + totalDist \cdot rd$ `// Sampled point in 3D space.`
    $height \leftarrow terrainMap(samplePoint.x, samplePoint.z)$
    $distance \leftarrow samplePoint.y - height$
    if $distance < epsilon$ **then**
        `// Sampled point is inside the terrain. Intersection was found.`
        $intersectionDist \leftarrow totalDist$
        return $intersectionDist$
    end
    $totalDist \leftarrow a \cdot distance$
    if $totalDist > farPlane$ **then**
        `// Ray reached far plane. There is no intersection.`
        return -1
    end
    $steps \leftarrow steps + 1$
    $epsilon \leftarrow epsilon \cdot totalDist$
end
return -1 `// Maximal number of steps was evaluated.`

Kapitola 4

Návrh riešenia

Cieľom tejto práce je vytvorenie grafického dema na vykresľovanie procedurálne generovanej krajiny vo fragment/pixel shaderi v jednom vykresľovacom prechode. To znamená, že vytvorené demo bude vykonávané na grafickej karte paralelne pre každý pixel. V tejto kapitole je popísaný návrh riešenia vykresľovania procedurálne generovanej krajiny bez konkrétnych implementačných detailov. Implementačné detaily tejto práce budú bližšie popísané v kapitole 5.

Vykresľovanie procedurálnej krajiny v tejto práci sa skladá z troch hlavných častí:

1. **Vykresľovanie terénu** – nájdenie priesečníka s terénom a výpočet osvetlenia a tieňov
2. **Vykresľovanie atmosféry** – výpočet farby oblohy a aplikovanie vzdušnej perspektívy
3. **Vykresľovanie oblakov** – vykreslenie tvaru oblakov a výpočet osvetlenia

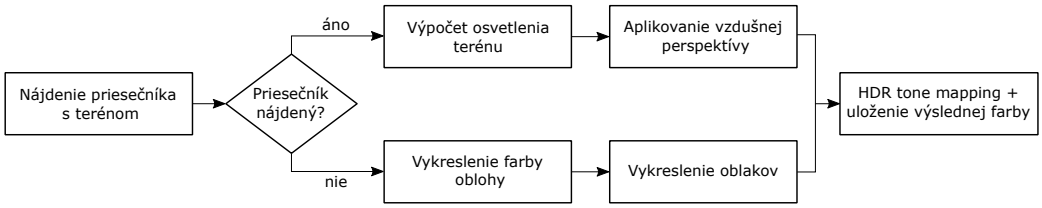
V tejto kapitole je popísaný spôsob riešenia každej z týchto častí a to v sekciách 4.2, 4.3 a 4.4. Najskôr je však v sekcii 4.1 popísaný vykresľovací reťazec.

4.1 Vykresľovací reťazec

Poradie vykresľovania jednotlivých častí krajiny je z hľadiska výkonu a vzhľadu výsledku dôležité. Pri vykresľovaní na grafickej karte je potrebné dávať pozor na vetvenie a na cykly s dopredu neznámym počtom iterácií, ktoré spôsobujú divergenciu vlákien GPU [8]. Vetveniu a cyklom sa pri vykresľovaní krajiny nie je možné vyhnúť ale správnou dekompozíciou väčších úloh na menšie a správnym poradím ich vykonávania je možné divergenciu minimalizovať. Rozklad hlavných častí riešenia na podčasti je bližšie popísaný v nasledujúcich sekciách.

Zjednodušený popis vykresľovania je nasledujúci: Ako prvé je potrebné vykresliť terén. Na terén sa potom aplikuje vzdušná perspektíva. Tam kde sa nenachádza terén sa vykreslí obloha a oblaky. Na výslednú farbu sa aplikuje HDR *tone mapping* a gamma korekcia. Zjednodušený diagram vykresľovacieho reťazca je zobrazený na obrázku 4.1.

Pre zjednodušenie diagramu sa predpokladá, že kamera sa vždy nachádza pod oblakmi a žiadne oblaky sa nenachádzajú medzi kamerou a terénom.



Obr. 4.1: Diagram vykresľovacieho reťazca na vykresľovanie procedurálne generovanej krajiny. Tento diagram je len náčrt ako by mal vykresľovací reťazec logicky vyzerať. Skutočný reťazec je v dôsledku minimalizácie divergencie trochu iný. Zmeny sú popísané v priebehu tejto kapitoly.

4.2 Terén

Na vykresľovanie terénu bol zvolený upravený algoritmus *raymarchingu* popísaný v sekcii 3.3. Tento algoritmus je vhodný na vykreslenie terénu a aj tieňov. Keďže spomínaný algoritmus je iteratívny s dopredu neznámym počtom iterácii, vzniká pri ňom divergencia vlákien GPU. Na jej minimalizáciu je vhodné úlohu vykreslenia terénu rozdeliť na menšie podúlohy vykonávané sekvenčne. Príklad takéhoto rozdelenia je: nájdenie priesečníka s terénom, výpočet tieňov a výpočet osvetlenia. Porovnanie vysoko-divergentného a optimalizovaného algoritmu je v tabuľke 4.1.

Algoritmus 5: Vykresľovanie terénu s vysokou mierou divergencie.

```
1 while steps < maxSteps do
2 |   ...
3 |   if intersection then
4 |   |   shadow ← getShadow(...)
5 |   |   color ← shade(...)
6 |   |   return color
7 |   end
8 |   ...
9 end
```

Algoritmus 6: Vykresľovanie terénu s minimalizovanou divergenciou.

```
1  while steps < maxSteps do
2  |   ...
3  |   if intersection then
4  |   |   break
5  |   end
6  |   ...
7  end
8  ...
9  shadow ← getShadow(...)
10 color ← shade(...)
11 return color
```

Tabuľka 4.1: Porovnanie dvoch algoritmov na vykresľovanie terénu pomocou *raymarchingu*. V algoritme 5 sa tiene a osvetlenie počítajú v cykle hľadania priesečníka s terénom. Funkcia *getShadow()* počíta tiene rovnakým iteračným algoritmom s dopredu neznámym počtom iterácií. Vysoká divergencia je spôsobená tým, že rôzne pixely končia v rôznych iteráciách oboch cyklov. Minimalizácia divergencie v algoritme 6 je dosiahnutá presunutím výpočtu tieňov a osvetlenia mimo cyklus *raymarchingu*. Takáto úprava dokáže výrazne znížiť čas vykresľovania.

4.2.1 Generovanie terénu

Terén v tejto práci je reprezentovaný guľou, na ktorú je premietnutá výšková mapa. Guľový terén umožní zobraziť zakrivenie planéty, čo spolu s guľovou atmosférou vyzerá realistickejšie ako by to bolo v prípade terénu reprezentovaného rovinou. Vyššie spomenutá výšková mapa má tvar funkcie, ktorej vstupy sú 2D súradnice a jej výstup je výška terénu v danom mieste. Výhoda takéhoto prístupu je veľmi vysoké rozlíšenie výškovej mapy, ktoré umožní vysokú mieru detailu. Nevýhoda je, že táto funkcia je vyhodnocovaná vždy keď je potrebné získať výšku terénu – v prípade *raymarchingu* použitého v tejto práci to je v každej iterácii tohto algoritmu. Výšková mapa je generovaná postupmi popísanými v sekcii 2.3. Ako bolo spomenuté, tieto algoritmy sú založené na iteratívnom skladaní šumov so znižujúcou sa frekvenciou a amplitúdou.

Generovanie šumu

Šum, ktorý je použitý v algoritmoch na generovanie výškovej mapy terénu musí byť opakovateľný. To znamená, že tento šum musí byť vo forme funkcie, ktorá pre rovnaké vstupy (súradnice) vráti vždy rovnakú hodnotu. Takúto funkciu je možné vytvoriť hashovaním alebo iným spôsobom generovania pseudonáhodných čísel [3] Spôsob použitý v tejto práci je popísaný v sekcii 5.3. Ďalšia požadovaná vlastnosť tohto šumu je, že musí byť hladký (nemal by obsahovať príliš vysoké frekvencie). Toto je dosiahnuté aplikovaním dolno-priepustného filtru na šum vytvorený z pseudonáhodných čísel. Dva typy šumov s vyššie popísanými vlastnosťami použité v tejto práci sú:

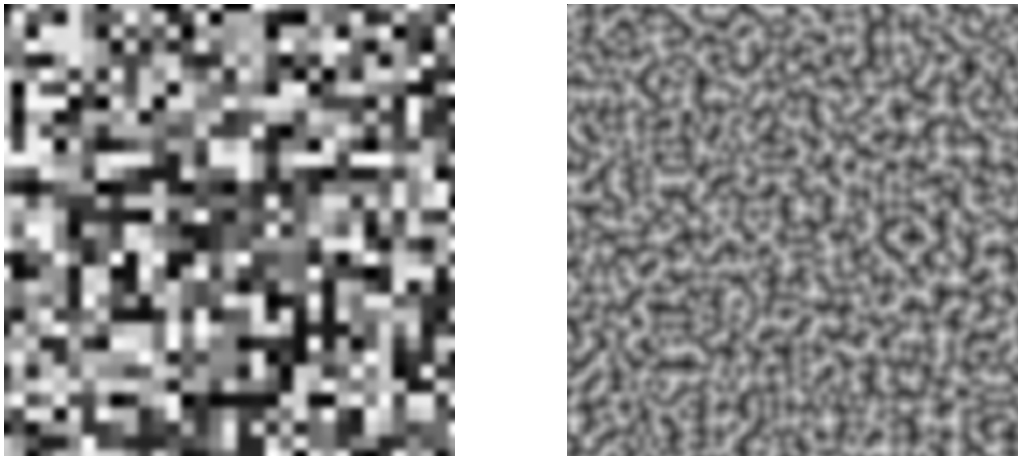
1. **Hodnotový šum (*value noise*)** – Jeho základ je n-dimenzionálna mriežka pseudonáhodných čísel. Hodnotový šum sa získa interpoláciou 2^n najbližších buniek mriežky z okolia požadovanej súradnice. Výsledný šum obsahuje určité artefakty, ktoré sú však pri sčítaní dostatočného počtu oktáv tohto šumu takmer neviditeľné. Výhoda hodnotového šumu je jeho jednoduchosť a rýchlosť výpočtu.

2. **Gradientný šum (*gradient noise*)** – Jeho základ je n-dimenzionálna mriežka n-dimenzionálnych pseudonáhodných jednotkových vektorov (gradientov). Prvý krok pri výpočte gradientného šumu je nájdenie 2^n najbližších buniek mriežky v okolí bodu, pre ktorý sa šum počíta. Následne sa pre každú z týchto buniek spočíta vektor vedúci z bunky do bodu pre ktorý sa šum počíta. Ďalej sa pre tieto štyri vektory spočíta skalárny súčin s hodnotou gradientu v ich prislúchajúcej bunke. Výsledok sa získa interpoláciou týchto hodnôt. Jedna z implementácii tohto šumu je Perlinov šum.

Na obrázku 4.2 sú porovnané 2D verzie týchto šumov.

4.2.2 Osvetlenie terénu

Osvetlenie terénu v tejto práci bolo inšpirované osvetlením popísaným v článku od Iniga Quileza [17]. V tejto práci bolo cieľom vytvoriť realistické osvetlenie, ktoré nie je náročné na výkon. Hlavný zdroj svetla vo vonkajšom prostredí je Slnko, ale veľká časť tohto svetla je ambientné a nepriame svetlo. Keďže vytvárané demo je vykresľované v jednom prechode fragment shaderom, techniky ako globálna iluminácia na výpočet nepriameho osvetlenia alebo realistický výpočet ambientného osvetlenia sú neprijateľné. Tieto efekty je však pri vonkajších scénach možné veľmi dobre aproximovať pomocou jednoduchých trikov, pretože správanie ambientného a nepriameho svetla je predvídateľné. Každá zložka svetla je reprezentovaná jedným smerovým svetlom so špecifickou intenzitou, smerom a farbou.



(a) Hodnotový šum (b) Gradientný šum

Obr. 4.2: Porovnanie hodnotového a gradientného šumu.

Priame osvetlenie

Táto časť osvetlenia je tvorená priamym svetlom zo Slnka. Jeho intenzita je zo všetkých troch svetiel najvyššia a je to jediné svetlo, ktoré vrhá tiene. Tiene sú počítané rovnakým algoritmom akým sa vykresľuje terén, ale lúč sa vyšle z osvetľovaného bodu smerom k primárnemu svetelnému zdroju. Ak sa nájde priesečník s terénom, osvetľovaný bod je v tieni. Tento algoritmus je možné jednoducho modifikovať aby pomocou neho bolo možné vykresľovať mäkké tiene bez zvýšenej výpočtovej náročnosti. Keďže tento algoritmus v každej iterácii počíta odhad vzdialenosti od terénu (viď algoritmus 4), jednoducho sa uloží najnižšia vzdialenosť zo všetkých iterácii a podľa tejto vzdialenosti sa vytvorí mäkký tieň [19]. Čím bližšie k terénu sa lúč priblíži, tým je tieň tmavší.

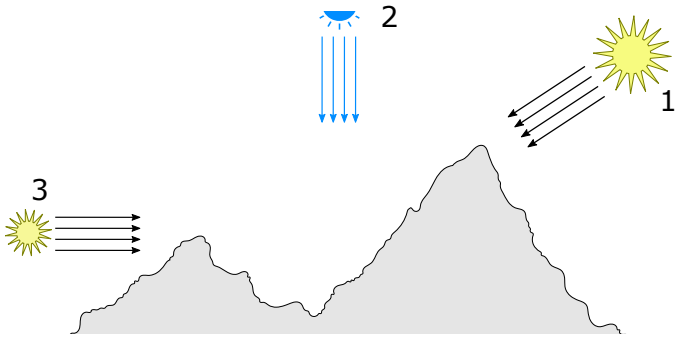
Ambientné osvetlenie

Ambientná časť osvetlenia je svetlo prichádzajúce do osvetľovaného bodu zo všetkých okolitých smerov. Vo vonkajších scénach je to hlavne slnečné svetlo, ktoré bolo rozptýlené atmosférou. Správny spôsob jeho počítania je zozbieranie tohto svetla v pologuli okolo osvetľovaného bodu. Ako bolo spomenuté vyššie, tento postup je z dôvodu výpočtovej náročnosti neprijateľný. Miesto toho je táto zložka aproximovaná smerovým svetlom, ktoré sa nachádza priamo nad terénom a svieti kolmo nadol. Toto svetlo má farbu oblohy a jeho intenzita je výrazne nižšia ako intenzita priameho svetla. Na túto zložku by mohlo byť aplikované zatienenie okolím (*ambient occlusion*), ktoré však v tejto práci nie je počítané.

Nepriame osvetlenie

Za nepriame osvetlenie je vo vonkajších scénach považované hlavne slnečné svetlo odrazené od povrchu terénu, ktoré osvetľuje iný povrch. Tento typ osvetlenia sa typicky počíta globálnou iluináciou. Pre potreby tejto práce sa môže predpokladať, že slnečné svetlo sa od povrchu odrazí do smeru, z ktorého prišlo. Toto správanie je možné aproximovať smerovým svetlom nachádzajúcim sa na opačnej strane ako Slnko, nasmerovaným horizontálne k terénu. Prakticky to znamná, že jeho x a z súradnice sú voči Slnku otočené a jeho súradnica y je nula. Intenzita tohto svetla je trochu vyššia ako intenzita ambientného svetla

a jeho farba je zmiešaná farba priameho svetla a terénu. Zatienenie okolím by malo byť aplikované rovnako aj na toto osvetlenie. Na obrázku 4.3 sú znázornené smery všetkých popísaných smerových svetliel, ktoré tvoria osvetlenie terénu.



Obr. 4.3: Znázornenie smerových svetiel v scéne, ktoré reprezentujú priame (1), ambientné (2) a nepriame osvetlenie (3). Ambientné osvetlenie je modré svetlo s nízkou intenzitou, ktoré aproximuje modré svetlo prichádzajúce z oblohy. Nepriame osvetlenie predstavuje priame svetlo, ktoré bolo odrazené od terénu. Preto smeruje horizontálne, opačným smerom ako primárne svetlo.

Osvetľovací model

Na výpočet osvetlenia je použitý Lambertov osvetľovací model s difúznymi odrazmi. Tento model je postačujúci, pretože väčšina materiálov terénu nevyžaduje spekulárne odlesky. Intenzita svetla v osvetľovanom bode je pri Lambertovom osvetlení závislá len na kosínuse uhla medzi normálou povrchu a vektorom k zdroju svetla. Na výpočet osvetlenia povrchu jedným svetlom sa používa rovnica (4.1).

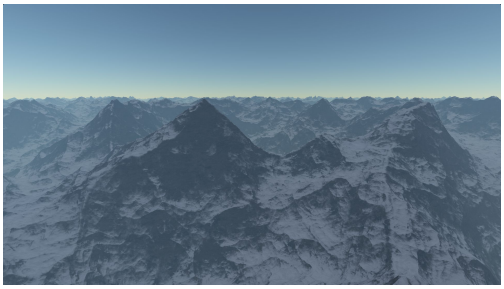
$$I \ = (N \cdot L) \circ I_L \tag{4.1}$$

kde I je intenzita osvetlenia povrchu, N je normála a I_L je intenzita svetelného zdroja. Operátor $\cdot$ označuje skalárny súčin vektorov a operátor $\circ$ označuje súčin vektorov po zložkách. Rovnica (4.1) sa použije na výpočet každej zložky svetla a výsledné osvetlenie je rovné ich súčtu. Na získanie osvetlenej farby terénu je tento výsledok nutné vynásobiť po zložkách s farbou materiálu v danom bode. Na obrázku 4.4 sú znázornené jednotlivé zložky svetla.

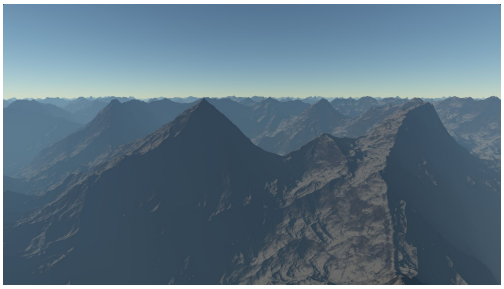
Výpočet normál

Na výpočet osvetlenia je nevyhnutné poznať normály povrchu. Keďže terén je definovaný ako dvojrozmerná výšková funkcia, na výpočet normál je možné použiť derivácie tejto funkcie. Derivácie sa jednoducho vypočítajú pomocou centrálnych diferencií [2]: $f(\boldsymbol{p} + \frac{1}{2}\boldsymbol{h}) - f(\boldsymbol{p} - \frac{1}{2}\boldsymbol{h})$, kde $f(\boldsymbol{p})$ je funkcia výškovej mapy a $\boldsymbol{p}$ je bod v rovine x-z. Vektor $\boldsymbol{h} = (\varepsilon, 0)$ alebo $\boldsymbol{h} = (0, \varepsilon)$, kde ε reprezentuje presnosť výpočtu. Pri $\boldsymbol{h} = (\varepsilon, 0)$ centrálna diferencia aproximuje parciálnu deriváciu v smere osi x. Rovnako pri $\boldsymbol{h} = (0, \varepsilon)$ aproximuje parciálnu deriváciu v smere osi z [2]. Nenormalizovaná normála sa získa ich vektorovým súčinom. Na výpočet normály v bode $\boldsymbol{p}$ je teda potrebné vypočítať výšky terénu v štyroch priľahlých bodoch znázornených na obrázku 4.5. Vzorec na výpočet normály $\vec{N}$ v bode $\boldsymbol{C}$ pomocou

(a) Priame



(b) Ambientné



(c) Nepriame

(d) Priame + ambientné + nepriame

Obr. 4.4: Znázornenie jednotlivých zložiek osvetlenia. Ambientné osvetlenie osvetľuje celý terén rovnomerne. Nepriame osvetlenie predstavuje svetlo odrazené od povrchov osvetlených priamym osvetlením. Preto táto zložka osvetľuje hlavne povrchy odvrátené od zdroja priameho osvetlenia.

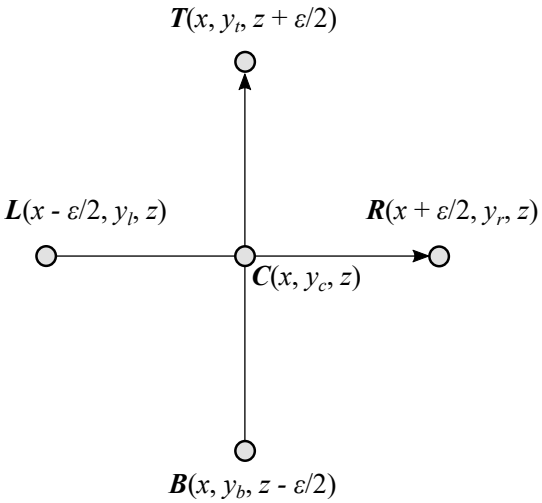
vyššie popísaného postupu je zobrazený v rovnici 4.2. Všetky body v tejto rovnici sú znázornené na obrázku 4.5.

$$\vec{N} = normalize((\boldsymbol{R} - \boldsymbol{L}) \times (\boldsymbol{T} - \boldsymbol{B})) \tag{4.2}$$

4.3 Atmosféra

Pod vykresľovanie atmosféry spadajú dve úlohy: výpočet farby oblohy a výpočet vzdušnej perspektívy. Výpočet farby oblohy sa vykonáva pre pixely kde nebol nájdený priesečník s terénom. Výpočet vzdušnej perspektívy sa naopak vykonáva pre pixely kde bol priesečník nájdený. Takéto vetvenie v shaderi prináša ďalšiu divergenciu. Pri porovnaní rovníc na výpočet farby oblohy (rovnica (3.20)) a výpočet vzdušnej perspektívy (rovnica (3.25)) je zjavné, že rovnice sú takmer rovnaké. Obe obsahujú integráciu priameho osvetlenia zo Slnka a rovnica na výpočet vzdušnej perspektívy obsahuje len jeden člen navyše. Tento člen reprezentuje farbu terénu (ktorá bola vypočítaná v predchádzajúcom kroku vykresľovania) vynásobenú celkovou priehľadnosťou medzi pozorovateľom a terénom. Túto priehľadnosť je možné akumulovať počas integrácie priameho osvetlenia, takže tieto dve úlohy sa môžu zjednotiť a tým sa odstráni divergencia. Ako bolo popísané v sekcii 3.2, integrácia sa vypočíta *raymarchingom*.

Posledný rozdiel medzi dvoma riešenými úlohami je, že pri výpočte farby oblohy sa integruje slnečné svetlo medzi kamerou a okrajom atmosféry, pričom pri výpočte vzdušnej perspektívy to je medzi kamerou a priesečníkom s terénom. Tento problém sa pri *raymarchingu* vyrieši jednoducho. Pri výpočte farby oblohy sa za maximálnu vzdialenosť *raymarchingu* zvolí vzdialenosť priesečníka s okrajom atmosféry, ktorý je reprezentovaný guľou



Obr. 4.5: Vizualizácia centrálnej diferencie na výpočet normál terénu. Normála sa získa vektorovým súčinom vektorov vzniknutých zo štyroch vzoriek výškovej mapy. Súradnice y_l, y_r, y_t, y_b sa vypočítajú vyhodnotením výškovej mapy v danom bode. Konštanta ε reprezentuje presnosť výpočtu.

okolo terénu. Pri výpočte vzdušnej perspektívy to zase bude vzdialenosť priesečníka s terénom. Výsledný algoritmus na vykresľovanie atmosféry je zobrazený v algoritme 7. Tento algoritmus implementuje výpočet rovnice (3.24). Konštanty *scatteringCoeffR/M* a *extinctionCoeffR/M* sú hodnoty koeficientov rozptylu a extinkčných koeficientov Rayleighovho a Mieho rozptylu vo výške $0\,m$, spomenuté v sekciách 3.2.2 a 3.2.3.

V prípade, že sa algoritmom 7 počíta vzdušná perspektíva aplikovaná na terén, po výpočte je potrebné správne zmiešať farbu terénu s vypočítanou farbou atmosféry. Na výpočet tejto operácie slúži rovnica (4.3).

$$color = terrainColor \cdot rayTransmittance + skyColor \tag{4.3}$$

kde *color* je výsledná farba po aplikovaní vzdušnej perspektívy na terén s farbou $terrainColor$. Hodnoty $rayTransmittance$ a $skyColor$ sú hodnoty vypočítané algoritmom 7.

Algoritmus 7: Vykresľovanie atmosféry pomocou *raymarchingu*. Algoritmus založený na článku [21].

Input: Primary ray origin ro and normalized direction rd, normalized light direction $lightDir$, distance to intersection with the terrain $intersectionDist$ or maximal possible value if there is no intersection with the terrain, maximal number of primary steps $maxSteps$ and secondary steps $maxStepsLight$, $nearPlane$, Rayleigh and Mie extinction and scattering coefficients $extinctionCoeffR/M$, $scatteringCoeffR/M$

Output: color of the sky $skyColor$, total transmittance along primary ray $rayTransmittance$

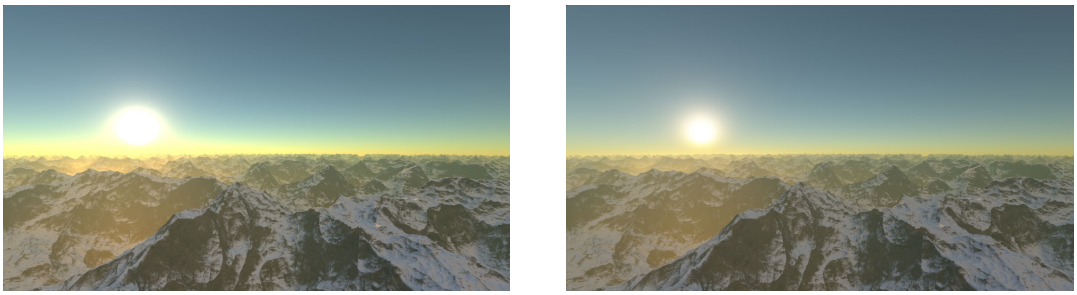
$atmosphereIntersectionDist \leftarrow getAtmosphereIntersectionDistance(ro, rd)$
if $atmosphereIntersectionDist < intersectionDist$ **then**
    `// There is no intersection with the terrain`
    $intersectionDist \leftarrow atmosphereIntersectionDist$
end
$stepLength \leftarrow (intersectionDist - nearPlane)/maxSteps$
$totalDist \leftarrow nearPlane$
$cosDir \leftarrow rd \cdot (-lightDir)$ `// cosine of an angle between two vectors`
$rayleighSum, mieSum \leftarrow (0, 0, 0)$
$opticalDepthR, opticalDepthM \leftarrow 0, 0$
$phaseR \leftarrow computePhaseR(cosDir)$ `// Rayleigh phase function value`
$phaseM \leftarrow computePhaseM(cosDir)$ `// Mie phase function value`
for $steps \leftarrow 0$ **to** $maxSteps$ **do**
    $samplePoint \leftarrow ro + (totalDist + stepLength/2) \cdot rd$
    $height \leftarrow getHeight(samplePoint)$
    $stepOpticalDepthR \leftarrow e^{-height/scaleHeightR} \cdot stepLength$
    $stepOpticalDepthM \leftarrow e^{-height/scaleHeightM} \cdot stepLength$
    $opticalDepthR \leftarrow opticalDepthR + stepOpticalDepthR$
    $opticalDepthM \leftarrow opticalDepthM + stepOpticalDepthM$
    $opticalDepthLightR \leftarrow raymarchToLightR(samplePoint, -lightDir)$
    $opticalDepthLightM \leftarrow raymarchToLightM(samplePoint, -lightDir)$
    $tau \leftarrow extinctionCoeffR \cdot (opticalDepthR + opticalDepthLightR) + extinctionCoeffM \cdot (opticalDepthM + opticalDepthLightM)$
    $stepTransmittance \leftarrow e^{-tau}$
    $rayleighSum \leftarrow rayleighSum + stepTransmittance \cdot stepOpticalDepthR$
    $mieSum \leftarrow mieSum + stepTransmittance \cdot stepOpticalDepthM$
    $totalDist \leftarrow totalDist + stepLength$
end
$rayTransmittance \leftarrow e^{-(scatteringCoeffR \cdot opticalDepthR + extinctionCoeffM \cdot opticalDepthM)}$
$skyColor \leftarrow (rayleighSum \cdot scatteringCoeffR \cdot phaseR + mieSum \cdot scatteringCoeffM \cdot phaseM) \cdot sunIntensity$

4.3.1 HDR vykresľovanie

Vykresľovanie atmosféry nevyzerá veľmi dobre bez HDR[1] vykresľovania, pretože použité rovnice generujú hodnoty s vysokým rozsahom. Hodnoty v okolí Slnka sú príliš vysoké

[1] High Dynamic Range – vysoký dynamický rozsah

a v iných miestach oblohy sú hodnoty oveľa nižšie. Z tohto dôvodu je na výslednú farbu pixelu aplikovaný HDR *tone mapping*. Je to operácia, ktorá prevedie farby z priestoru s vysokým dynamickým rozsahom do priestoru s nízkym dynamickým rozsahom. V tejto práci je použitý exponenciálny *tone mapping*, ktorý na prevod hodnôt používa exponenciálnu funkciu: $1 - e^{-exposure \cdot color}$. Konštanta *exposure* simuluje citlivosť kamery – nižšie hodnoty vytvárajú tmavší obraz a vyššie hodnoty zase svetlejší obraz. Použitá exponenciálna funkcia prevedie hodnoty do intervalu $\langle 0; 1 \rangle$. Dôležitosť vykresľovania s HDR *tone mappingom* je demonštrovaná na obrázku 4.6.

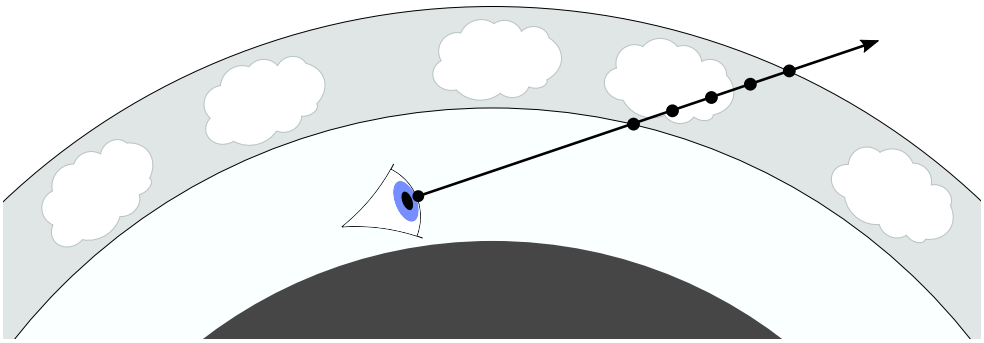


(a) vykresľovanie bez HDR *tone mappingu* (b) vykresľovanie s HDR *tone mappingom*

Obr. 4.6: Porovnanie vykresľovania s HDR *tone mappingom* a bez. Najväčší rozdiel je vidieť v okolí Slnka, ktorého tvar nie je možné na obrázku naľavo rozoznať.

4.4 Oblaky

Posledná časť vykresľovania je vykresľovanie oblakov. Oblaky sú participujúce médium, takže ich vykresľovanie je založené na princípoch z kapitoly 3.1. Pre zjednodušenie ich vykresľovania sa oblaky nachádzajú len v presne definovanom priestore nad terénom. Tento priestor je ohraničený dvomi sústrednými guľami s rozdielnym polomerom. Táto konfigurácia je zobrazená na obrázku 4.7.



Obr. 4.7: Obrázok zobrazuje princíp vykresľovania oblakov a ich umiestnenie v scéne. Vrstva, v ktorej sa nachádzajú oblaky je vyfarbená svetlo-šedou farbou. Takáto konfigurácia uľahčí nájdenie počiatočného a koncového bodu pri vykresľovaní *raymarchingom*.

4.4.1 Tvar oblakov

V scéne nie sú definované jednotlivé oblaky so svojou vlastnou pozíciou a inými vlastnosťami. Miesto toho je vrstva, v ktorej sa nachádzajú oblaky, vyplnená 3-rozmerným *fBm* šumom. Hodnota šumu je v intervale $\langle 0; 1\rangle$ a táto hodnota je interpretovaná ako hustota oblačnosti v danom mieste. Hustota s hodnotou nula znamená, že v danom mieste sa nenachádzajú žiadne čiastočky tvoriace oblak.

Tento prístup funguje dobre a je jednoduchý. Jeho nevýhoda však je, že nevytvára úplne realistické tvary oblakov. Oblaky v prírode vyzerajú ako stúpajúca para, takže majú oblé, zakrivené tvary na vrchnej strane. Tento efekt však použitou technikou nie je možné dosiahnuť.

4.4.2 Algoritmus vykresľovania

Cieľom vykresľovania oblakov je vyriešenie rovnice (3.14). Na to sa rovnako ako pri atmosfére použije *raymarching*, ktorý bol popísaný v sekcii 3.1.4. Najskôr sa nájdu priesečníky lúča vychádzajúceho z kamery s obidvomi guľami, ktoré ohraničujú vrstvu oblakov (viď obrázok 4.7). Následne sa medzi týmito priesečníkmi v niekoľkých bodoch pozdĺž lúča vzorkuje hustota oblakov zo šumu, ktorá je použitá na výpočet koeficientu rozptylu a extinkčného koeficientu vo danom bode. V každom vzorkovanom bode sa rovnako počíta priame osvetlenie, tiež *raymarchingom* (viď obrázok 3.8). Po vykreslení oblakov je na ne, rovnako ako na terén, potrebné aplikovať vzdušnú perspektívu.

4.4.3 Fázová funkcia

Pri výpočte osvetlenia oblakov je použitá dvojitá Henyey-Greensteinova fázová funkcia popísaná v sekcii 3.1.2. Táto funkcia simuluje predný a aj zadný rozptyl svetla, čo oproti obyčajnej Henyey-Greensteinovej funkcii pomôže hlavne keď sa Slnko nachádza za kamerou (viď obrázok 4.8).

(a) Predný a zadný rozptyl (b) Iba predný rozptyl

Obr. 4.8: Porovnanie oblakov vykreslených s použitím dvojitej Henyey-Greensteinovej fázovej funkcie (vľavo) a obyčajnej Henyey-Greensteinovej fázovej funkcie (vpravo). Slnko sa nachádza za kamerou, takže aby sa svetlo v oblakoch rozptýlilo dozadu smerom do kamery, je potrebné simulovať aj zadný rozptyl.

Kapitola 5

Implementácia

V tejto kapitole sú popísané niektoré dôležité a zaujímavé implementačné detaily tejto práce. Práca je vytvorená v nástroji *Shadertoy*[1]. Je to online komunita a nástroj, ktorý umožňuje vytvárať a zdieľať shadere v jazyku GLSL[2]. *Shadertoy* zabezpečuje vykresľovanie v internetovom prehliadači pomocou WebGL. Používateľ má však prístup len k fragment shaderu, pomocou ktorého vykresľuje obrázok na kresliace plátno.

5.1 Vykresľovanie vo fragment shaderi

Vykresľovanie a generovanie scény vo fragment shaderi so sebou prináša niekoľko obmedzení. Jedno z najväčších obmedzení je chýbajúca vstupná geometria. Geometriu je teda potrebné generovať vždy za behu shaderu.

Pri vykresľovaní vo fragment shaderi sa vytvára shader, ktorý je vykonávaný paralelne pre každý pixel. Aby sa pre každý pixel nevykonával úplne rovnaký kód, vykresľovanie je potrebné parametrizovať súradnicou konkrétneho pixelu/fragmentu. Keďže celé implementované demo je vykresľované *raymarchingom*, každý pixel sa líši smerom lúča vyslaného z kamery smerom do scény. Tento lúč je použitý na nájdenie priesečníku s terénom, na vykreslenie atmosféry a aj oblakov.

5.2 Implementovaná scéna

Základ scény tvorí guľa, ktorá reprezentuje planétu. Na túto guľu je aplikovaná výšková mapa, ktorá modifikuje polomer planéty čím sa vytvára terén. Výšková mapa je generovaná pomocou ryhovaného multifraktálu popísaného v sekcii 2.3.2, v ktorom je použitý 2D gradientný šum. Okolo planéty sa nachádza vrstva atmosféry, ktorej okraj je vymedzený obalovou guľou. V atmosfére je v určitej výške nad terénom umiestnená vrstva oblakov, ktorá je vyplnená šiestimi oktávami 3D hodnotového šumu. Tento šum reprezentuje hustotu oblakov.

Kamera je umiestnená nad terénom a hýbe sa po rovnej trajektórii, pričom mení svoju výšku aby sa prispôsobila výške terénu v danom mieste. Na uľahčenie výpočtu pozície kamery a na odstránenie problémov, ktoré by mohli vzniknúť pri premietnutí procedurálnej výškovej mapy na guľovú planétu, je pohyb kamery len zdanlivý. Kamera v skutočnosti stojí na mieste, mení sa len jej výška, a pohybuje sa výšková mapa terénu a oblaky.

[1] https://www.shadertoy.com/
[2] https://www.khronos.org/opengl/wiki/OpenGL_Shading_Language

5.3 Generovanie šumu

Šum je základný stavebný prvok tejto práce. Používa sa pri generovaní terénu, textúr a aj oblakov. Za šum je v tomto texte považovaná n-rozmerná mriežka vyplnená náhodnými číslami. Ako bolo spomenuté v sekcii 4.2.1, šum musí byť implementovaný vo forme funkcie, ktorá pre rovnaké vstupy vytvorí vždy rovnaký výstup. Z tohto dôvodu je na generovanie náhodných čísel použitá hashovacia funkcia, ktorá aplikuje na vstup určité matematické operácie, ktorými je vytvorený výstup.

5.3.1 Generovanie náhodných čísel

V tejto práci je na generovanie náhodných čísel použitá hashovacia funkcia z rodiny funkcií PCG. Táto funkcia poskytuje veľmi dobrý kompromis medzi rýchlosťou a kvalitou [6]. Funkcia je zobrazená vo výpise 5.1.

```
uint hashPCGu(uint x)
{
  uint state = x * 747796405u + 2891336453u;
  uint word = ((state >> ((state >> 28u) + 4u)) ^ state) * 277803737u;
  return (word >> 22u) ^ word;
}
```

Výpis 5.1: Hashovacia funkcia z rodiny funkcií PCG, ktorá generuje 32-bitové hodnoty v intervale $\langle 0; UINT\_MAX \rangle$. Implementácia prevzatá z [6].

Táto funkcia pracuje v doméne neznamienkových celých čísel a generuje hodnoty v intervale $\langle 0; UINT\_MAX \rangle$. Pri generovaní šumu v počítačovej grafike sa však väčšinou pracuje so znamienkovými desatinnými číslami a požadované generované hodnoty sú v intervale $\langle 0; 1 \rangle$ alebo $\langle -1; 1 \rangle$. Vstupy a výstupy tejto funkcie je preto potrebné vhodne upraviť. Znamienkové desatinné čísla na vstupe sú prevedené na neznamienkové aplikovaním absolútnej hodnoty. Následne sú zaokrúhlené nadol aby sa odstránila desatinná časť a toto číslo je potom pretypované na `uint` (viď výpis 5.2). Výstup hashovacej funkcie je prevedený do intervalu $\langle 0; 1 \rangle$ vydelením maximálnou hodnotou, ktorá je v tomto prípade $UINT\_MAX = 4\,294\,967\,295$.

5.3.2 Generovanie viacdimenzionálneho šumu

V predchádzajúcej sekcii bola predstavená funkcia na generovanie náhodných čísel, ktorá môže byť použitá na vytvorenie jednorozmerného šumu. Na generovanie terénu je však potrebný 2D šum a na generovanie oblakov 3D šum. Na to je potrebné aby hashovacia funkcia akceptovala viacdimenzionálne vstupy. Tento problém sa dá riešiť niekoľkými spôsobmi. Najjednoduchší postup je použiť lineárnu kombináciu vstupných súradníc X, Y, Z vynásobených prvočíslami a, b, c (viď rovnica (5.1)) [6].

$$hash(aX + bY + cZ) \tag{5.1}$$

Táto metóda je rýchla ale generuje opakujúce sa vzory. Tie je možné eliminovať použitím iného prístupu na transformáciu vstupných súradníc. Tento prístup používa vnorené hashovacie funkcie, a je zobrazený v rovnici (5.2) [6].

$$hash(X + hash(Y + hash(Z))) \tag{5.2}$$

Prístup s vnorenými hashovacími funkciami generuje kvalitné výsledky bez opakovania vzorov ale za cenu zvýšeného výpočtového času, pretože hashovacia funkcia sa musí vyhodnotiť

viackrát. Z tohto dôvodu nie je táto technika použitá. Pri generovaní viacdimenzionálneho šumu je použitá technika zobrazená v rovnici (5.3). Je to technika založená na lineárnej kombinácii ale namiesto súčtu vstupných súradníc používa operáciu XOR. Generované výsledky sú kvalitnejšie ako pri použití lineárnej kombinácie [6].

$$hash(aX \wedge bY \wedge cZ) \tag{5.3}$$

Vo výpise 5.2 je zobrazená funkcia použitá na generovanie 2D šumu pomocou PCG hashovacej funkcie.

```
float hashPCG(vec2 x)
{
  x += 3250000.0;
  x = abs(x);
  uvec2 p = uvec2(floor(x));
  return float(hashPCGu(149u*p.x ^ 233u*p.y)) / UINT_MAX;
}
```

Výpis 5.2: Príklad funkcie na generovanie 2D šumu. Používa funkciu `hashPCGu()` z výpisu 5.1. Na prvom riadku funkcie sa vstupný bod posunie aby zrkadlenie vzniknuté aplikovaním absolútnej hodnoty nebolo viditeľné v okolí počiatku súradnicovej sústavy.

Postupmi popísanými v tejto sekcii sa generuje len základný biely šum. Tento šum je následne použitý pri generovaní hodnotového a gradientného šumu.

5.4 Výpočet normál terénu

V sekcii 4.2.2 bol popísaný spôsob akým sa počítajú normály terénu. Tento výpočet je možné optimalizovať a tým urýchliť výpočet. Optimalizovaný postup je zobrazený vo výpise 5.3. V porovnaní s rovnicou (4.2) tento postup nepočíta s bodmi ale len s výškami terénu.

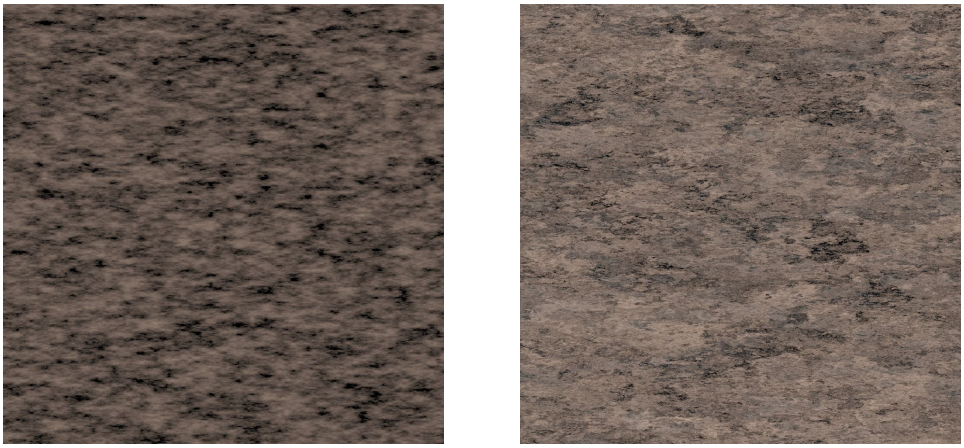
```
vec3 getNormal(vec2 p, float epsilon)
{
  float leftHeight   = f(p.x-epsilon, p.z);
  float rightHeight  = f(p.x+epsilon, p.z);
  float bottomHeight = f(p.x, p.z-epsilon);
  float topHeight    = f(p.x, p.z+epsilon);
  vec3 N = vec3( leftHeight - rightHeight,
                 2.0*epsilon,
                 bottomHeight - topHeight );
  return normalize(N);
}
```

Výpis 5.3: Optimalizovaný výpočet normál terénu metódou centrálnych diferencií [16]. Funkcia `f()` počíta výškovú mapu terénu v danom bode, `epsilon` definuje presnosť výpočtu a `p` je súradnica výškovej mapy, pre ktorú sa počíta normála.

5.5 Textúrovanie

Ryhovaný multifraktál, ktorý je použitý na generovanie terénu vytvára terén pripomínajúci skalnaté pohorie. Základnú farbu terénu preto tvorí kamenná textúra. V shaderi sú implementované dva postupy na generovanie 3D kamennej textúry, medzi ktorými je možné prepínať (popísané v sekcii 5.6). Prvý postup je zameraný na generovanie kvalitnej textúry s veľkou mierou detailu. Tento postup je výpočtovo náročný, pretože používa štyri volania

funkcie na výpočet *fBm* šumu. Druhý postup je viac zameraný na výkon a používa len jedno volanie funkcie na výpočet *fBm* šumu. Porovnanie oboch textúr je zobrazené na obrázku 5.1.



(a) Výkon (b) Kvalita

Obr. 5.1: Porovnanie dvoch kamenných textúr implementovaných v tejto práci. Textúra vľavo je generovaná postupom zameraným na výkon. Textúra vpravo je generovaná postupom zameraným na kvalitu.

Okrem kamennej textúry je na terén aplikovaná aj snehová textúra. Táto textúra je veľmi jednoduchá textúra bielej farby. Výber textúry záleží na sklone svahu v danom textúrovanom mieste. Sklon svahu je počítaný ako prevrátená hodnota súradnice y normály terénu. Prechod medzi textúrami je počítaný funkciou `smoothstep()` z jazyka GLSL. Aby tento prechod vyzeral reálnejšie, prvý parameter funkcie `smoothstep()` je modifikovaný *fBm* šumom, ktorý tomuto prechodu pridá na komplexnosti a náhodnosti.

5.5.1 Filtrovanie textúr

Fraktálny šum, ktorý je použitý pri procedurálnom generovaní textúr, vytvorí v obraze vysoké frekvencie a tým prinesie aliasing. Tento problém je riešený filtrovaním textúr. Najskôr sa zistí akú veľkú plochu v scéne zaberá fragment, pre ktorý je textúra počítaná. Keďže demo je implementované vo fragment shaderi, táto plocha môže byť vypočítaná použitím funkcií `dFdx()` a `dFdy()`, pomocou ktorých sa zistí vzdialenosť textúrovaných bodov medzi susednými fragmentmi. Následne sa vypočíta niekoľko vzoriek procedurálnej textúry vo vypočítanom priestore, ktorý fragment zaberá. Výsledná farba textúry sa vypočíta ako aritmetický priemer všetkých vzoriek.

5.6 Ovládanie

Parametre vykresľovania implementovaného dema sú ovládané direktívami `#define`. Definovaním makra `LOW_QUALITY` sa zníži kvalita vykresľovania a zvýši sa výkon. Toto makro zmení hodnotu niekoľkých parametrov naraz, avšak používateľ môže meniť aj hodnoty jednotlivých makier, ktoré menia parametre vykresľovania. V nasledujúcom zozname sú makrá, ktoré slúžia na ladenie výkonu a kvality; za ich názvom je krátky popis a v zátvorke je typ ich hodnoty:

- `MAX_RENDERING_DISTANCE` – maximálna vykresľovacia vzdialenosť (`float`)
- `MAX_MARCHING_STEPS` – maximálny počet krokov *raymarchingu* pri vykresľovaní terénu (`int`)
- `MAX_MARCHING_STEPS_SHADOW` – maximálny počet krokov *raymarchingu* pri vykresľovaní tieňov terénu (`int`)
- `TEXTURE_LOW_QUALITY` – použije sa kamenná textúra s nízkou kvalitou (`0` alebo `1`)
- `TEXTURE_FILTERING` – zapne sa filtrovanie textúr (`0` alebo `1`)
- `FILTERING_SAMPLES` – počet vzoriek textúry v jednej dimenzii použitých pri filtrovaní textúr; celkový počet vzoriek je teda zadané čislo na druhú (`int`)
- `ATMOS_STEPS` – počet primárnych krokov *raymarchingu* pri vykresľovaní atmosféry (`int`)
- `ATMOS_LIGHT_STEPS` – počet sekundárnych krokov *raymarchingu* pri vykresľovaní atmosféry (`int`)
- `CLOUD_STEPS` – počet primárnych krokov *raymarchingu* pri vykresľovaní oblakov (`int`)
- `CLOUD_LIGHT_STEPS` – počet sekundárnych krokov *raymarchingu* pri vykresľovaní oblakov (`int`)
- `CLOUD_COVERAGE` – pokrytie oblohy oblačnosťou (`float` v intervale $\langle 0; 1 \rangle$)

Najväčší vplyv na výkon majú počty sekundárnych krokov *raymarchingu* atmosféry a oblakov, pretože to je počet krokov, ktorý sa vykoná v každom primárnom kroku. Okrem týchto parametrov je možné ešte meniť parametre, ktoré upravujú hlavne vzhľad a nevplývajú príliš na rýchlosť vykresľovania. Najdôležitejšie z nich sú vypísané v nasledujúcom zozname:

- `OCTAVES` – počet oktáv šumu použitých pri vykresľovaní terénu (`int`)
- `OCTAVES_SHADOW` – počet oktáv šumu použitých pri vykresľovaní tieňov terénu (`int`)
- `OCTAVES_NORMAL` – počet oktáv šumu použitých pri výpočte normál terénu (`int`)
- `SMOOTH_RIDGES` – zapne vyhladzovanie ostrých hrán terénu (`0` alebo `1`)
- `CLOUDS_HEIGHT` – výška oblakov (`float`)
- `CLOUDS_THICKNESS` – hrúbka oblakov (`float`)
- `CAM_HEIGHT` – výška kamery nad terénom (`float`)
- `CAM_SPEED` – rýchlosť pohybu kamery (`float`)

Ďalšie parametre, ktoré je možné meniť a neboli popísané v tejto sekcii sú napríklad pozícia Slnka na oblohe, výška a mierka terénu, koeficient rozptylu a absorpcie oblakov, hustota oblakov a parametre ich fázovej funkcie.

Kapitola 6

Vyhodnotenie

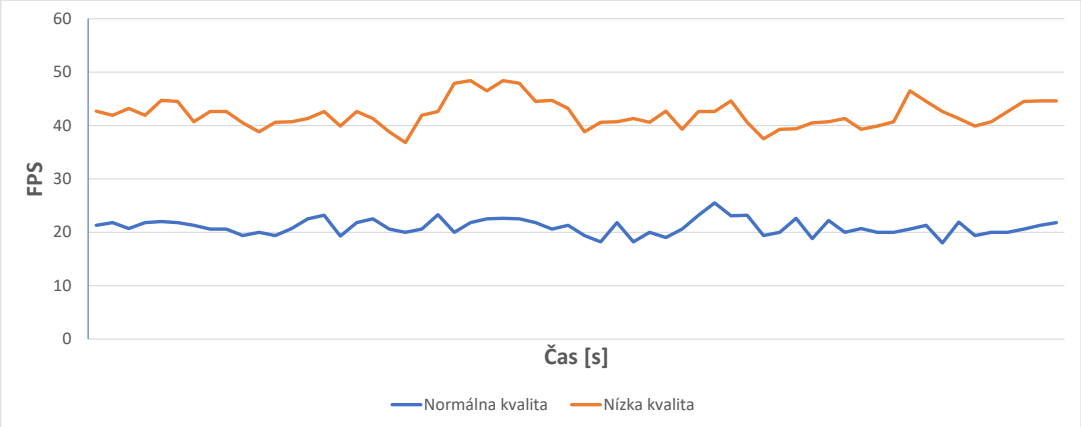
Táto kapitola je zameraná na vyhodnotenie výsledkov tejto práce. Nachádzajú sa tu merania výkonu a porovnanie vplyvu parametrov vykresľovania na výkon a výslednú kvalitu výstupu. Merania boli vykonané v prehliadači Mozilla Firefox 100.0.1 na grafickej karte NVIDIA GeForce GTX 1650 Ti s rozlíšením 800×450.

6.1 Porovnanie grafických nastavení

V shaderi sú predpripravené dve grafické nastavenia vykresľovania:

1. Normálna kvalita
2. Nízka kvalita

Predvolené nastavenie je Normálna kvalita. Nízka kvalita sa nastaví definovaním makra `LOW_QUALITY` (viď kapitola 5.6). Toto nastavenie vypne vykresľovanie oblakov a zníži celkovú kvalitu vykresľovania. Graf porovnania počtu snímok za sekundu je zobrazený na obrázku 6.1.



Obr. 6.1: Porovnanie počtu snímok za sekundu oboch grafických nastavení implementovaného dema. Graf znázorňuje prvých 60 sekúnd dema.

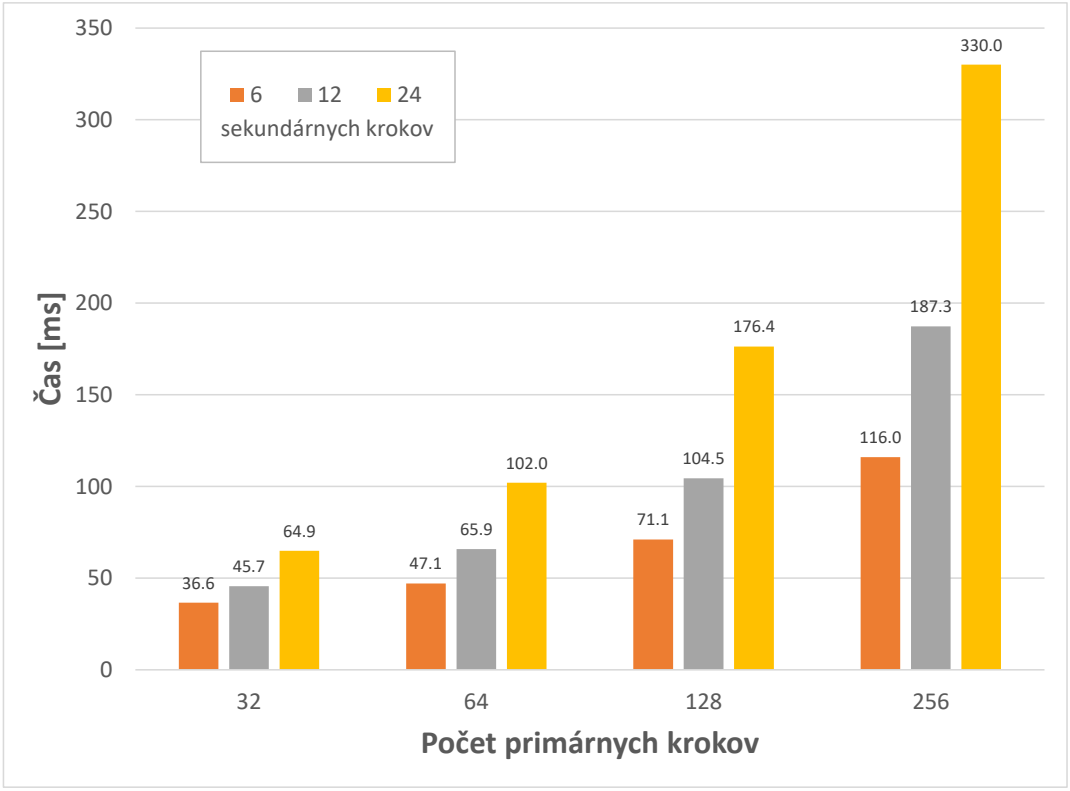
Ako je možné vidieť na obrázku 6.1, nastavenie Nízka kvalita dosahuje dvojnásobné počty snímok za sekundu. Hlavný dôvod je vypnutie vykresľovania oblakov, pretože táto časť vykresľovania je veľmi náročná na výpočet.

V grafe je možné vidieť, že počty snímok za sekundu sú v celom priebehu celkom stabilné, pretože komplexnosť scény sa takmer nemení. Pri zohľadnení rozlíšenia použitého pri meraní (800×450), sú dosiahnuté počty snímok celkom nízke. Je to z dôvodu výpočtovej náročnosti použitých metód a rovnako aj tým, že všetok obsah dema (terén, textúry, atmosféra, oblaky) je generovaný v reálnom čase pre každý vykresľovaný pixel.

6.2 Vplyv počtu krokov raymarchingu na výkon

Najväčší vplyv na výkon majú počty krokov *raymarchingu* pri vykresľovaní oblakov. V tejto sekcii je preto porovnaný vplyv počtu krokov na výkon a v nasledujúcej sekcii je porovnaný vplyv počtu krokov na kvalitu výsledku.

Pri vykresľovaní oblakov je možné nastaviť počty dvoch typov krokov: primárnych a sekundárnych. Za primárne kroky sú považované kroky na primárnom lúči. Tento lúč vychádza z kamery smerom do scény. V každom primárnom kroku sa na výpočet osvetlenia vyšle smerom k svetelnému zdroju sekundárny lúč (viď obrázok 3.8). Počet sekundárnych krokov je teda počet krokov pri *raymarchingu* sekundárneho lúča. Ná obrázku 6.2 je zobrazený graf priemerného času vykreslenia jedného snímku pre rôzne počty primárnych a sekundárnych krokov. Tieto časy boli merané na scéne so statickou kamerou, kde obloha zaberá približne dve tretiny obrazu (viď obrázok 6.4).

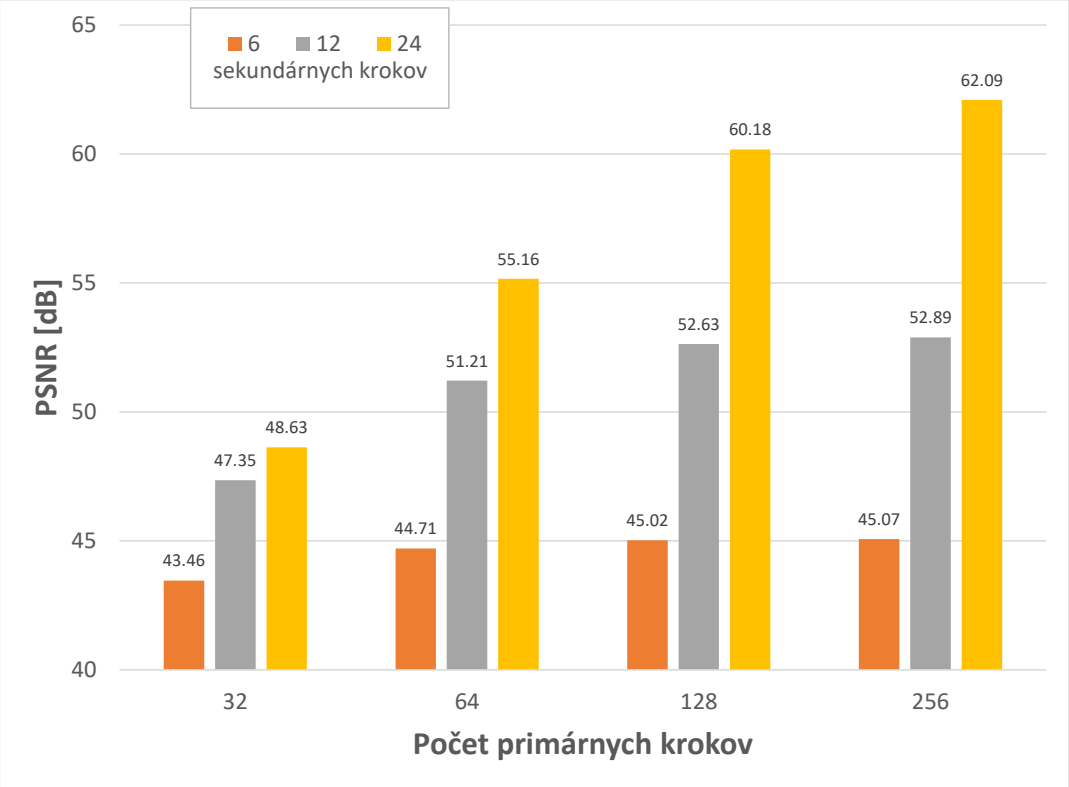


Obr. 6.2: Porovnanie počtu krokov *raymarchingu* oblakov na priemerný čas vykresľovania jedného snímku.

6.3 Vplyv počtu krokov raymarchingu na výsledný obraz

Na zistenie vplyvu počtu krokov *raymarchingu* na kvalitu výsledného obrazu bola použitá metrika PSNR (*Peak signal-to-noise ratio*). Táto metrika predstavuje pomer medzi maximálnou možnou energiou signálu a energiou šumu[1].

V tomto teste bola použitá rovnaká scéna ako pri meraní v predchádzajúcej sekcii (viď obrázok 6.4). Z tejto scény boli zhotovené snímky v rozlíšení 1920×1080 pre rôzne počty krokov a tieto snímky boli následne porovnané s referenčnou snímkou pomocou metriky PSNR. Referenčná snímka bola vykreslená s použitím 512 primárnych a 32 sekundárnych krokov. Výsledok porovnania je zobrazený v grafe na obrázku 6.3.



Obr. 6.3: Porovnanie vplyvu počtu krokov *raymarchingu* oblakov na kvalitu výsledného obrazu pomocou metriky PSNR (*Peak signal-to-noise ratio*). Na vykreslenie referenčného obrázku bolo použitých 512 primárnych a 32 sekundárnych krokov.

Na grafe je možné vidieť, že pre rovnaké počty sekundárnych krokov nie sú medzi rôznymi počtami primárnych krokov príliš veľké rozdiely. Najväčšie rozdiely sú medzi počtami primárnych krokov pri 24 sekundárnych krokoch. Podľa použitej metriky sú výsledky s 6 sekundárnymi krokmi výrazne menej kvalitné ako výsledky s väčším počtom sekundárnych krokov. Pri porovnaní tohto grafu s grafom v obrázku 6.2, môže byť konštatované, že najlepší pomer medzi kvalitou a výpočtovým časom má vykresľovanie s 12 sekundárnymi krokmi a 32 alebo 64 primárnymi krokmi. Pri vykresľovaní s len 32 primárnymi krokmi (obrázok 6.6) sú však viditeľné pruhové artefakty, hlavne v tenších častiach oblakov. Tieto artefakty vznikajú kvôli malému počtu krokov a je možné ich eliminovať náhodným posunom počiatku

[1] https://cs.wikipedia.org/wiki/PSNR

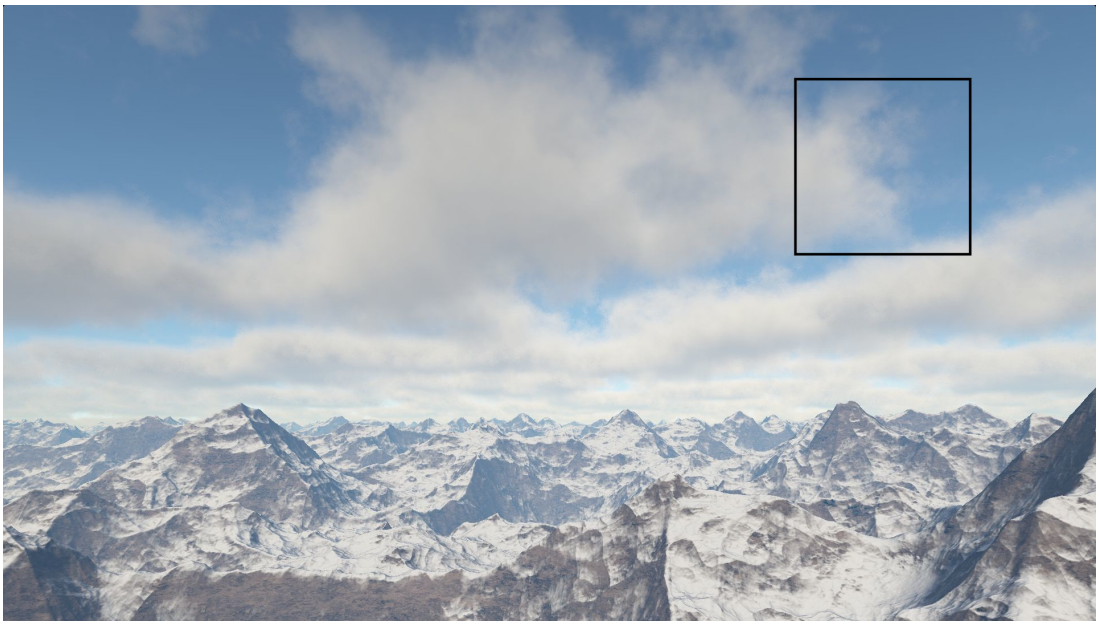
primárneho lúču *raymarchingu* (viď obrázok 6.7). Tento posun však do oblakov vnesie šum, ktorý je viditeľný hlavne v tenkých častiach oblakov.

Obrázok vytvorený s použitím 64 primárnych a 12 sekundárnych krokov už vyzerá takmer na nerozoznanie od referenčného obrázku (viď obrázok 6.5). Na porovnanie, priemerný čas vykresľovania s týmto počtom krokov je $65,9\,ms$ a pre referenčný obrázok to je $869,6\,ms$. Najlepší pomer medzi kvalitou a časom vykresľovania sa zdá byť pri tomto počte krokov.

Obr. 6.4: Referenčný obrázok vykreslený s použitím 512 primárnych a 32 sekundárnych krokov.

Obr. 6.5: Obrázok vykreslený s použitím 64 primárnych a 12 sekundárnych krokov.

Obr. 6.6: Obrázok vykreslený s použitím 32 primárnych a 12 sekundárnych krokov. Zvýraznené sú pruhové artefakty, ktoré vznikajú pri použití nízkeho počtu krokov.



Obr. 6.7: Obrázok vykreslený s použitím 32 primárnych a 12 sekundárnych krokov. Pruhové artefakty, ktoré sú v obrázku 6.6 boli odstránené náhodným posunutím počiatku primárneho lúča pre každý pixel. Týmto však vznikol šum, ktorý je možné vidieť v zvýraznenej časti obrázku.

Kapitola 7

Záver

Cieľom tejto diplomovej práce bolo vytvoriť grafické demo procedurálne generovanej krajiny vo fragment shaderi. Na dosiahnutie tohto cieľa bolo potrebné naštudovať zložitú teóriu ako napríklad fraktály, šírenie svetla v médiu, šírenie svetla atmosférou, volumetrické vykresľovanie, atď. Naštudovaná teória bola popísaná v prvých dvoch kapitolách tejto práce. V nasledujúcich kapitolách bol popísaný návrh riešenia, zaujímavé implementačné detaily a vyhodnotenie.

Implementované demo obsahuje vykresľovanie procedurálne generovaného terénu, generovanie procedurálnych textúr, simuláciu realistickej atmosféry a vykresľovanie oblakov. Terén je generovaný pomocou ryhovaného multifraktálu, ktorý pripomína kamenné pohorie. Na jeho vykreslenie je použitý upravený algoritmus *raymarchingu*.

Výstupy implementovaného dema vyzerajú zaujímavo a realisticky. Jediný nedostatok generovaného terénu je jeho jednotvárnosť. Najväčšia nevýhoda implementovaného dema je vysoká výpočtová náročnosť.

Demo bolo vytvorené v online nástroji *Shadertoy*, kde má k nemu prístup široká verejnosť a môže poslúžiť ako návod alebo zdroj inšpirácie pre ostatných používateľov.

Literatúra

[1] Akenine-Möller, T., Haines, E., Hoffman, N., Pesce, A., Iwanicki, M. et al. Volumetric and Translucency Rendering. In: *Real-Time Rendering*. 4. vyd. A K Peters/CRC Press, 2018, kap. 14. ISBN 9780429225406.

[2] Cozzi, P. a Ring, K. Rendering Height Maps. In: *3D Engine Design for Virtual Globes*. 1. vyd. A. K. Peters, Ltd., 2011, kap. 11.2. ISBN 1568817118.

[3] Ebert, D., Musgrave, F., Peachey, D., Perlin, K., Worley, S. et al. *Texturing and Modeling: A Procedural Approach*. 3. vyd. Morgan Kaufmann Publishers, 2003. ISBN 1-55860–848-6.

[4] Hillaire, S. Physically Based Sky, Atmosphere and Cloud Rendering in Frostbite. In: *Physically Based Shading in Theory and Practice*. 2016. SIGGRAPH 2016. Dostupné z: https://www.ea.com/frostbite/news/physically-based-sky-atmosphere-and-cloud-rendering.

[5] Jarosz, W. *Efficient Monte Carlo Methods for Light Transport in Scattering Media*. San Diego, USA, 2008. Dizertačná práca. University of California. Dostupné z: https://cs.dartmouth.edu/wjarosz/publications/dissertation/.

[6] Jarzynski, M. a Olano, M. Hash Functions for GPU Rendering. *Journal of Computer Graphics Techniques (JCGT)*. October 2020, zv. 9, č. 3, s. 20–38. ISSN 2331-7418. Dostupné z: http://jcgt.org/published/0009/03/02/.

[7] Keinert, B., Schäfer, H., Korndörfer, J., Ganse, U. a Stamminger, M. Enhanced Sphere Tracing. In: Giachetti, A., ed. *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*. The Eurographics Association, 2014. DOI: 10.2312/stag.20141233. ISBN 978-3-905674-72-9. Dostupné z: http://dx.doi.org/10.2312/stag.20141233.

[8] Khronos. *Shader: Execution model and divergence* [online]. Október 2019 [cit. 25.4.2022]. Dostupné z: https://www.khronos.org/opengl/wiki/Shader#Execution_model_and_divergence.

[9] Kniss, J., Premoze, S., Hansen, C., Shirley, P. a McPherson, A. A Model for Volume Lighting and Modeling. In:. Máj 2003, sv. 9, s. 150–162. DOI: 10.1109/TVCG.2003.1196003. Dostupné z: https://ieeexplore.ieee.org/document/1196003.

[10] Mandelbrot, B. *The Fractal Geometry of Nature*. 2. vyd. Times Books, 1982. ISBN 0716711869.

[11] Mayaux, B. *Real-Time Volumetric Rendering* [online]. 2013 [cit. 21.2.2022]. Dostupné z: http://patapom.com/topics/Revision2013/Revision%202013%20-%20Real-time%20Volumetric%20Rendering%20Course%20Notes.pdf.

[12] Murray, S. *Building Worlds Using Math(s)*. 2017. Game Developers Conference. Dostupné z: https://www.gdcvault.com/play/1024514/Building-Worlds-Using.

[13] Nishita, T., Sirai, T., Tadamura, K. a Nakamae, E. Display of the Earth Taking into Account Atmospheric Scattering. In: *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*. New York, USA: Association for Computing Machinery, 1993, s. 175–182. SIGGRAPH '93. ISBN 0897916018. Dostupné z: https://doi.org/10.1145/166117.166140.

[14] O'Neil, S. Accurate Atmospheric Scattering. In: *GPU Gems 2*. 1. vyd. Addison-Wesley Professional, 2005, kap. 16. ISBN 9780321545411.

[15] Premože, S. *Light Transport in Participating Media* [online]. 2003 [cit. 27.2.2022]. Dostupné z: http://renderwonk.com/publications/s2003-course/premoze1/notes-premoze.pdf.

[16] Quilez, I. *Raymarching Terrains* [online]. 2002 [cit. 30.3.2022]. Dostupné z: https://www.iquilezles.org/www/articles/terrainmarching/terrainmarching.htm.

[17] Quilez, I. *Outdoors Lighting* [online]. 2013 [cit. 28.4.2022]. Dostupné z: https://iquilezles.org/articles/outdoorslighting/.

[18] Quilez, I. *Fractional Brownian Motion* [online]. 2019 [cit. 3.12.2021]. Dostupné z: https://www.iquilezles.org/www/articles/fbm/fbm.htm.

[19] Quilez, I. *Soft Shadows in Raymarched SDFs* [online]. 2021 [cit. 12.5.2022]. Dostupné z: https://www.reedbeta.com/blog/hash-functions-for-gpu-rendering/.

[20] Žára, J., Beneš, B., Sochor, J. a Felkel, P. Procedurální modelování. In: *Moderní počítačová grafika*. 2. vyd. Computer Press, 2004, kap. 8. ISBN 80-251-0454-0.

[21] Scratchapixel. *Simulating the Colors of the Sky* [online]. [cit. 22.3.2021]. Dostupné z: https://www.scratchapixel.com/lessons/procedural-generation-virtual-worlds/simulating-sky/simulating-colors-of-the-sky.