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

ZOBRAZOVÁNÍ ATMOSFÉRICKÝCH JEVŮ V KRAJINĚ

VISUALISATION OF ATMOSPHERIC PHENOMENA IN TERRAIN

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JAN OLEXA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. TOMÁŠ MILET**
SUPERVISOR

BRNO 2017

Abstrakt

Cílem práce je navrhnout a implementovat grafickou aplikaci zobrazující atmosférické jevy v krajině, například déšť či mlhu. Program je napsán v jazyce C++ za využití knihovny OpenGL verze 4.0.

Abstract

The goal of this thesis is to design and implement a graphic application displaying atmospheric phenomena in terrain, such as rain or fog. The program is written in C++ language and uses the OpenGL library, version 4.0.

Klíčová slova

Atmosférické jevy, OpenGL, krajina, grafika, C++

Keywords

Atmospheric phenomena, OpenGL, terrain, graphics, C++

Citace

OLEXA, Jan. *Zobrazování atmosférických jevů v krajině*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Milet

Zobrazování atmosférických jevů v krajině

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Tomáše Mileta. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Olexa
16. května 2017

Poděkování

Rád bych zde poděkoval panu inženýru Tomáši Miletovi, který významě rozšířil mé znalosti v oblasti grafiky a byl vždy v dobré náladě.

Obsah

Kapitola 1

Úvod

Atmosférickými jevy rozumíme různé přírodní úkazy v atmosféře, jako jsou například déšť, sníh nebo mlha. Zkoumáním těchto jevů se zabývá především meteorologie, která nabyté poznatky využívá například k předpovídání počasí.

Cílem této bakalářské práce není vytvořit přesný fyzikální model těchto jevů, nýbrž pouze jejich vizuální zobrazení v počítačové grafice s pokud možno co nejvyšší estetickou hodnotou. K realizaci tohoto cíle jsou využity pokročilé techniky rozhraní OpenGL verze 4.0 a vyšší a programovací jazyk C++. Důraz je kladen především na procedurální generování těchto jevů, které docílí vyšší versatility programu.

Práce je tvořena několika kapitolami, které jsou logicky členěny dle tématu.

První část textu pojednává o teoretických postupech, které byly v různé míře využity pro tvorbu této práce či se jí logicky dotýkají. Druhá část vysvětluje různá rozhodnutí učiněná během návrhu práce a jejich vliv na výslednou podobu aplikace. Třetí část se zabývá detaily implementace jednotlivých částí programu. Celá práce je pak shrnuta v závěru.

Kapitola 2

Teorie

2.1 Atmosférické jevy

Atmosférickými jevy nazýváme nejrůznější úkazy v atmosféře a na zemském povrchu, ať už se jedná o déšť, mlhu, kroupy, či například blesk. Většinu z nich lze shrnout pod pojem meteory, jež se dále dělí podle složení a podmínek vzniku na hydrometeory, litometeory, fotometeory a elektrometeory.

Pod hydrometeory spadají vodní i ledové částečky, které padají nebo se vznáší v atmosféře, nebo jsou usazené na předmětech na zemi nebo ve volné atmosféře. Jedná se např. o déšť, mlhu či sníh.

Litometeory jsou lehké, z vody nepocházející částice rozptýlené ve vzduchu, např. prach, kouř nebo písek.

Fotometeory značí světelné jevy způsobené slunečním nebo měsíčním světlem, např. halové jevy, korona a duha.

Jako elektrometeory značíme viditelné a slyšitelné projevy atmosférické elektřiny. Jedná se např. o blesk, hřmění nebo polární záři.

V této bakalářské práci jsou implementovány atmosférické jevy déšť, sníh, mlha a soumrakové jevy. Informace čerpány z portálu *Techmania* [3].

2.2 Postupy generování terénu

Asi nejčastějším postupem generování terénu je použití výškových map – do programu jsou nahrána data z obrázku ve stupních šedi, intenzita jejichž pixelů určuje výslednou výšku terénu. Obvykle platí pravidlo že čím světlejší je pixel, tím vyšší bude bod terénu, přičemž délku a šířku terénu si může zvolit programátor. Nevýhodou tohoto postupu je, že jelikož je pro daný bod udána právě jedna výška, nelze jednoduše generovat plochy vertikální, či vytvořit převisy a jeskyně.

Další skupinou postupů tvorby terénu je využití fraktálů, tzn. soběpodobných, rekurzivně generovaných objektů. Postupy spadající do této kategorie se obvykle vyznačují vysokou flexibilitou, rychlostí a jednoduchostí implementace, ale také obtížností ovlivnit výsledný terén. Jako příklad je možno uvést techniku Triangle edge subdivision (volně přeloženo jako Dělení trojúhelníkových hran). Terén je zde rozdělen do čtvercových ploch které jsou každá následně rozdělena do dvou trojúhelníků. Každý trojúhelník je pak rozdělen do dalších čtyř. V dalších děleních je pak ke středu hrany přidána Gaussova náhodná veličina, jejíž hodnota je popsána rovnicí 2.1.

$$S = k \cdot 2^{-iH} \tag{2.1}$$

V rovnici 2.1 je H dimenze fraktálu, i úroveň iterace a k váhový faktor. Pokud je váhový faktor zvolen nulový, bude z procesu odstraněna veškerá náhodnost. Informace o výškových mapách a fraktálech čerpány z bakalářské práce Adama Juríka *Fractal Terrain Generation* [8].

Třetí typ postupů pro generování terénu spočívá v tzv. voxelech (analogie ke slovu pixel), které jsou základním prvkem objemových dat a reprezentují hodnotu svého objemu. Voxely jsou ukládány do trojrozměrné mřížky (na rozdíl od např. výškové mapy, kde se užívá mřížka dvojrozměrná), což umožňuje tvorbu nad a pod zemí, jako je např. převis či jeskyně. Vizualizace dat poté probíhá buď jejich transformací do dvourozměrného obrázku, nebo vykreslením pomocí geometrických primitiv, například krychlí. Informace o voxelové generaci terénu čerpány z diplomové práce Michala Černého *Generování terénu s využitím voxelové struktury* [6].

2.3 Phongův osvětlovací model

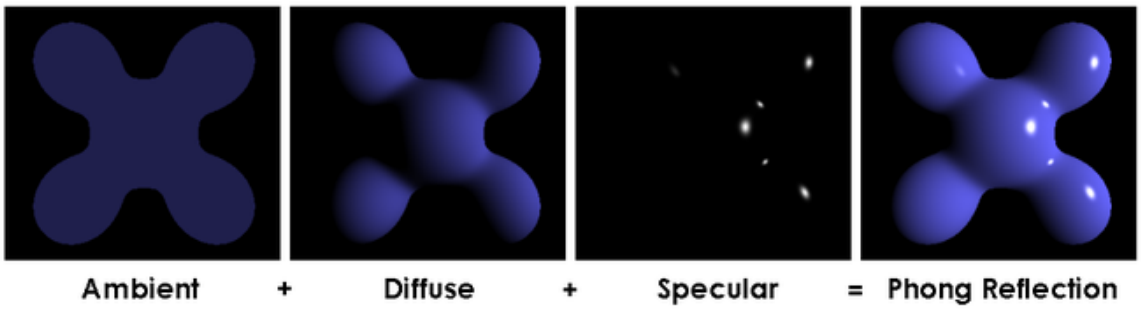
Phongův osvětlovací model je široce používaným modelem určeným k výpočtu odraženého světla z povrchu objektu. Světlo je v tomto modelu složeno ze tří světelných složek: ambientního světla, difúzního světla a odlesků.

Ambientní světlo dopadá na všechny části objektu rovnoměrně se stejnou intenzitou a ze všech směrů a je abstrakcí světla odraženého od ostatních předmětů. Reálná implementace odražených paprsků by byla výpočetně náročná a výsledný efekt by nebyl výrazně lepší. Ambientní světlo je tedy výpočetně levnou a efektivní alternativou, která zajišťuje, že plochy odvrácené od zdroje světla nebudou úplně černé, což by nebylo vizuálně atraktivní ani realistické. Čím vyšší je hodnota této složky, tím světlejší bude výsledná scéna.

Difúzní složka udává intenzitu světla odraženého od matného povrchu – světlo je zde odraženo všemi směry a jeho intenzita závisí na tom, jak natočený byl daný povrch vůči zdroji světla. Intenzita je vypočítána jakožto skalární součin normály povrchu a normalizovaného směrového vektoru paprsku světla. Pokud je tedy povrch natočený kolmo vůči paprsku, bude intenzita nejvyšší. Výsledný světelný efekt dodává scéně přirozeně vypadající stíny skrze poměrně jednoduchou implementaci. Informace o ambientní a difúzní složce čerpány z vlastní zkušenosti a ze slajdů přednášky Dr. Antona Gerledana [7].

Odlesky (ang. *specular light*) jsou světlo, které se od tělesa odráží převážně v jednom směru. Jsou nejpatrnější na hladkých površích a na rozdíl od ostatních složek závisí na pozici kamery. Těleso se v tomto případě chová jako ideální zrcadlo (tzn. úhel odrazu se rovná úhlu dopadu), díky čemuž lze snadno odvodit odrazový vektor světla od povrchu. Pak už je jen třeba spočítat úhel mezi tímto vektorem a vektorem mířícím od povrchu ke kameře. Pokud je tento úhel dostatečně malý (velikost závisí na konkrétní implementaci a lesklosti objektu), bude na tělese v daném místě vidět odlesk. Informace o odlescích čerpány ze slajdů z přednášek na Texaské Univerzitě ve městě Austin [4].

Ilustraci Phongova osvětlovacího modelu je možné si prohlédnout na obrázku 2.1.



Obrázek 2.1: Jednotlivé složky Phongova osvětlovacího modelu a obraz vzniklý jejich kombinací. Obrázek získán z: `https://upload.wikimedia.org/wikipedia/commons/thumb/6/6b/Phong_components_version_4.png/655px-Phong_components_version_4.png`

2.4 Generování náhodných čísel

Náhodnost je jednou z klíčových složek pro generování šumu, avšak také jednou z nejobtížněji získatelných. Pravá náhodnost je totiž v oblasti informatiky velice obtížně dosažitelná – počítač by k jejímu získání musel monitorovat náhodný jev reálného světa, což je řešení nákladné a pomalé. Z tohoto důvodu se v informatice používají tzv. pseudonáhodná čísla, to jest čísla která se ve skutečnosti řídí nějakým vzorem a náhodná nejsou, ale lidskému pozorovateli se tak jeví.

Funkcí schopných generovat pseudonáhodná čísla je celá řada, přičemž nejrozšířenější z nich jsou kongrugentní generátory, které se dále dělí do více podtypů. Jako příklad je možno uvést lineární kongrugentní generátor jehož matematický vzorec je uveden v rovnici 2.2.

$$x_n = (ax_{n-1} + b) \mod m \tag{2.2}$$

Proměnná a v rovnici 2.2 je člen multiplikativní, b aditivní a m modul. Jak je ze vzorce zřejmé, je pro výpočet dalšího náhodného čísla třeba znát číslo předchozí, a je tedy nutné definovat nějakou počáteční hodnotu. Této hodnotě se říká seed, neboli semeno. Informace o kongrugentních generátorech čerpány z přednáškových slajdů Petra Peringera přednášených na Vysokém Učení Technickém v rámci předmětu Modelování a Simulace [9]. Tato bakalářská práce využívá jakožto jednoduchý generátor pseudonáhodných čísel funkci popsanou matematickou rovnicí 2.3.

$$r = (\sin(x \cdot 12.9898 + y \cdot 78.233) \cdot 43758.5453) \mod 1 \tag{2.3}$$

V rovnici 2.3 značí r výsledné pseudonáhodné číslo a x a y jsou vstupy. Je možné si povšimnout, že tento jednoduchý generátor nevyužívá předchozí vygenerované hodnoty a nepotřebuje tedy seed, ale zato využívá dvou vstupů, což umožňuje poslat na vstup dvoudimenzionální vektor.

2.5 Perlinův šum

Jedním z nejčastějších cílů počítačové grafiky je vytvoření realisticky vypadajícího obrazu, ať už má být využit pro filmové efekty, animace, ilustrace či počítačové hry. Jednoduché geometrické útvary však něčeho takového dosáhnout nemohou – nedokážou popsat tvar

mraků, hor, či pobřeží bez výrazné abstrakce, která výslednou realističnost poškodí. Řešení tohoto problému nabízí fraktální geometrie, kde tzv. fraktály jsou objekty s na první pohled složitou strukturou, avšak jednoduchou matematickou definicí, kdy se v rámci fraktálu neustále opakují charakteristické tvary.

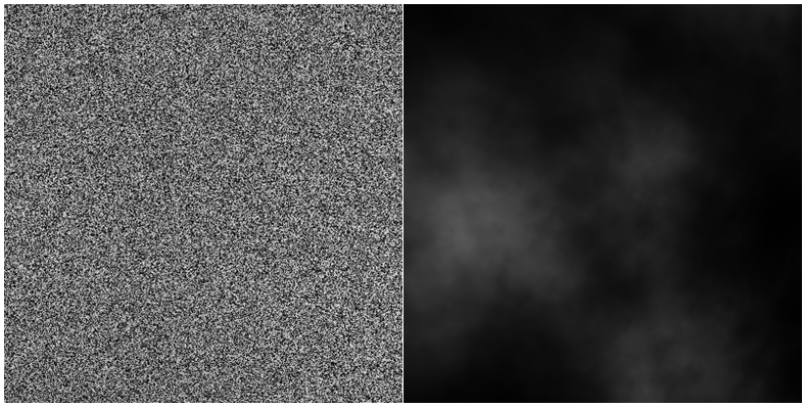
Použití složitějších geometrických tvarů však pro generaci realistického obrazu samotné nestačí – v přírodě zřídkakdy nalezneme zcela pravidelné útvary skládající se z perfektních fraktálů. Většina objektů reálného světa, jako je například koruna stromu či skalní útes, se skládá z tvarů si sobě podobných (např. podobně strukturované větve smrku), avšak nikoli zcela stejných (některé větve se rozbočují víckrát, než jiné, v jiných místech atp.). Tuto nepravidelnost je možno simulovat přidáním náhodnosti.

I v samotné náhodnosti však existuje problém – pokud z libovolných vstupů generuje zcela nahodilé výstupy, bude výsledný obraz vypadat eraticky.

Jedním z řešení všech těchto problémů je náhodná funkce s fraktální strukturou, Perlinův šum. V této funkci dostaneme ze stejného vstupu vždy stejný výstup, přičemž výstup je závislý na změnách vstupu, tzn. pokud bude hodnota vstupu měněna plynule, bude se plynule měnit i hodnota výstupu.

Samotný princip Perlinova šumu spočívá ve vytvoření několika šumových funkcí lišících se od sebe v intenzitě a měřítku. Perlinův šum pak získáme jejich složením (pořadí skládané funkce se nazývá oktáva), z čehož vyplývá, že se konkrétní výstup a implementace Perlinova šumu mohou výrazně lišit v závislosti na použitých funkcích. Informace o Perlinovu Šumu čerpány z vlastní zkušenosti a z bakalářské práce Jana Byšky *Perlinův šum a jeho aplikace* [5].

Názornou ukázku rozdílu mezi náhodným šumem a šumem Perlinovým ilustruje obrázek 2.2.



Obrázek 2.2: Rozdíl mezi šumem vytvořeným pomocí jednoduchého generátoru pseudonáhodných čísel (vlevo) a Perlinovým šumem (vpravo) je zřejmý už na první pohled.

2.6 Transform Feedback

Transform Feedback je jednou z pokročilých technik využívaných v OpenGL verze 3.0 a vyšších, která zachytává primitiva generovaná prací s vertexy a ukládá je do buffer objektů. To umožňuje zachování dat získaných během operací uvnitř vertex shaderu a použití jich ve více renderovacích cyklech bez nutnosti komunikace s procesorem.

Pro použití *transform feedback* je nutné nastavit patřičné parametry, jako například jména proměnných, jejichž hodnotu hodláme zapsat do buffer objektu, ještě před linková-

ním shaderového programu. Na konkrétní implementaci poté závisí, zda-li jsou nová data zapsaná do stejného buffer objektu, v jakém byla data vstupní, nebo jestli jsou zapsány do buffer objektu jiného. Informace ohledně této techniky byly čerpány z webového portálu Khronos Group [1].

V této bakalářské práci je techniky *transform feedback* využito při implementaci deště a sněžení.

2.7 Billboarding

Billboarding je název pro používání dvoudimenzionálních objektů ve trojdimenzionální grafice, které jsou vždy natočeny ke kameře. Využití nacházejí např. v počítačových hrách, kde mohou zastávat funkci ukazatele životů, textových bublin, apod. Implementace *billboardingu* není složitá a lze využít několik různých postupů. V OpenGL je však implementace obzvláště jednoduchá díky použití geometry shaderu. Pokud je totiž je totiž v geometry shaderu vzato primitivum (např. jeden vertex při bodovém vykreslování) a je transformováno na nějaké staticky dané primitivum (např. čtverec), pak toto výstupní primitivum bude vždy natočeno vůči kameře; nestane se, že by na takto vytvořený čtverec někdy bylo nahlíženo z boku.

Tato bakalářská práce využívá *billboarding* při implementaci sněhových vloček. Při tvorbě dešťových kapek se pak naopak snaží *billboardingu* co nejvíce zamezit (viz sekce 4.1). Informace o *billboardingu* byly čerpány z vlastní zkušenosti a z webu společnosti Microsoft [2].

Kapitola 3

Návrh

Cílem této kapitoly je objasnit různé skutečnosti a rozhodnutí v rámci projektu, která se přímo netýkají samotné impementace avšak která je vhodné zmínit.

3.1 Struktura kódu

Program využívá moderní techniky OpenGL a vysoké množství shaderů, které jsou logicky členěny do souborů podle toho, jaký jev mají vykreslovat. Na každý jev tak připadají dva až čtyři soubory se shadery, v závislosti na tom, zda je k vertex a fragment shaderům připojen i geometry shader či shader pro transform feedback. Funkce main() pak obsahuje nekonečnou smyčku – před smyčkou vytvoří všechny potřebné shader programy, nadefinuje a připraví veškeré vertex buffer objekty a vertex array objekty, načte textury a nadeklaruje potřebné proměnné. V programové smyčce tak skončí pouze příkazy nezbytné pro vykreslení jednotlivých atmosférických jevů a částí scény, které jsou logicky seskupeny k sobě. Výsledný kód tak dosahuje vysoké modularity, kdy není obtížné přidávat či odebírat nové atmosférické jevy z programu bez nechtěných zásahů do jeho jiných částí. Kód se tak také stává přehlednějším.

3.2 Scéna

Přestože s atmosférickými jevy přímo nesouvisí, byl vysoký důraz kladen i na samotnou krajinu a oblohu kolem ní. Toto rozhodnutí činí výstup esteticky pohlednější a skýtá místo pro další funkčnost, např. změny denní doby či pohyb zdroje světla v závislosti na pohybu slunce. Při výběru proporcí, barev a rychlosti průběhu jevů byl opět kladen důraz na vzhledovou stránku spíše než na fyzikální, ovšem s tou podmínkou, že se nebude nepřiměřeně vymykat reálnému světu. Mlžný opar tedy nebude disponovat širokou škálou pestrých barev, avšak například úsvit a soumrak a s nimi spojené červánky potrvají déle, než by v reálném světě vzhledem k délce dne měly, a budou mít velice výrazné, rudo-oranžové barvy.

3.3 Ovládání

Pro možnost inspekce libovolného bodu terénu (například pro prohlédnutí si mlhy z různých úhlů a vzdáleností) je nutné mít možnost pohybovat s kamerou. Ta se ovládá pomocí klávesnice a myši. Pohyb klávesami W (dopředu), A (doleva), S (dozadu) a D (doprava) byl

zvolen pro co největší komfort – v kombinaci s druhou rukou ovládající natočení kamery myší se jedná o ideální pozici, jak může potvrdit každý hráč počítačových her.

Jediným dalším ovládacím prvkem klávesa tabulátor, umístěná hned vedle kláves pro pohyb, která přepíná mezi počasím. Tři módy počasí jsou déšť, sníh a slunečno.

Kapitola 4

Implementace

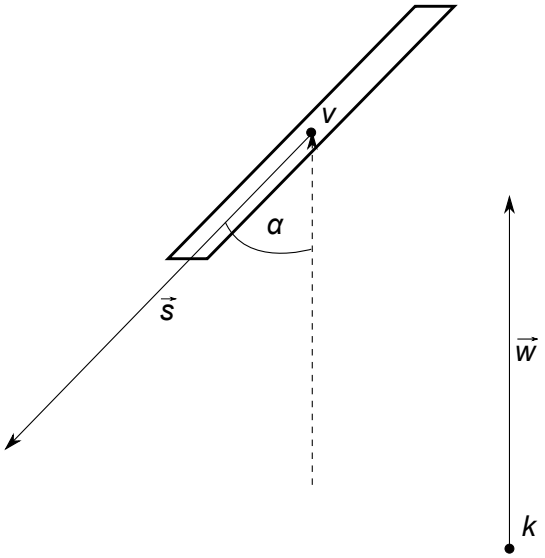
Zdrojový kód aplikace je vytvořen v prostředí aplikace Microsoft Visual Studio Express 2013 for Windows Desktop v programovacím jazyce C++. Program dále využívá knihovny GLEW, GLFW a FreeGLUT pro nahrávání rozšíření OpenGL a knihovnu GLM umožňující snazší práci s matematickými konstrukcemi. Některé části kódu byly převzaty od jiných autorů a následně upraveny pro účely této bakalářské práce. Veškeré tyto části jsou označeny v komentářích zdrojovém kódu spolu s odkazem ke zdroji.

V následujících podkapitolách bude rozebrána konkrétní implementace jednotlivých atmosférických jevů, terénu a pozadí.

4.1 Déšť a sníh

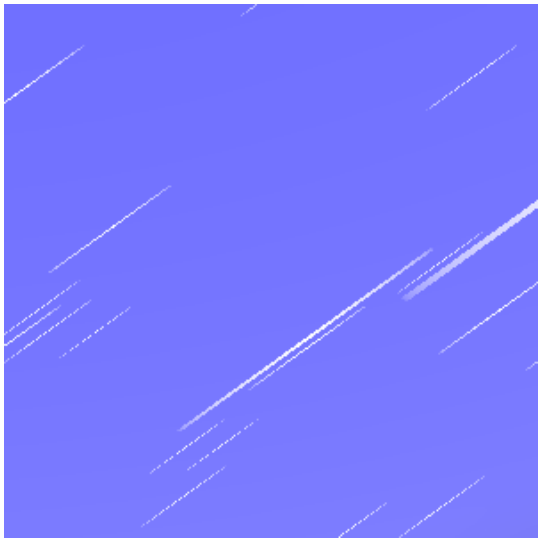
Déšť je implementován jakožto pole souřadnic v trojrozměrném prostoru generovaných jednoduchým generátorem pseudonáhodných čísel. Toto pole je před začátkem hlavní smyčky programu nahráno do vertex bufferu. V každém průchodu smyčkou jsou pak souřadnice o kousek posunuty v rámci techniky *transform feedback*, jež umožňuje posílání dat mezi shadery bez nutnosti průchodu skrze CPU, což snižuje nároky aplikace. Postupné posouvání těchto „kapek“ dodává dešti ve výsledné scéně dojem plynulého pohybu. Aby takto posouvané kapky nebyly posunuty příliš daleko od kamery a aby byla zachována iluze stálého deště po celé scéně, je nutné souřadnice ohraničit – pokud se jakákoli souřadnice dostane mimo pomyslnou krychli kolem kamery, je v dané ose posunuta na opačný konec krychle, načež pokračuje v pohybu. Kamera je tak vždy v samotném středu deště.

Poté, co se souřadnice dostanou do vertex shaderu je pro každou kapku spočítán úhel mezi jejím směrovým vektorem a vektorem pohledu kamery. Pomocí tohoto úhlu bude v geometry shaderu určena délka kapky (čím větší úhel, tím větší kapka), což dodá kapkám realističtější chování. Schéma kapky a vektorů ji ovlivňujících je možné vidět na obrázku 4.1.



Obrázek 4.1: Kapka je vytvořena v geometry shaderu kolem pozice vertexu v. Její tvar je dán jejím směrovým vektorem $\vec{s}$ a její délka úhlem α, který je mezi vektorem $\vec{s}$ a $\vec{w}$. Vektor $\vec{w}$ je vektor pohledu kamery začínající v její pozici k.

Geometry shader, mimo výše zmíněné délky, udává i šířku a sklon kapky, jež odvozuje z jejich směrového vektoru. Výsledný zkosený dvoudimenzionální obdélník vizualizující kapku deště je pak obarven ve fragment shaderu tak, aby jeho konce měly vyšší průsvitnost než jeho střed, což vede k lepšímu vzhledu kapky. Výsledná podoba dešťových kapek ve scéně je ilustrována obrázkem 4.2.



Obrázek 4.2: Dešťové kapky putují stejnosměrným rovnoměrným pohybem a jsou tedy vůči sobě rovnoběžné. Za povšimnutí stojí klesající intenzita bílé barvy na obou koncích kapky.

Sníh je řešen velice podobně dešti, také pomocí *transform feedback*, ale jelikož je vločka (při příslušném stupni abstrakce) malá koule, není nutné řešit úhel pohledu, z jakého na

ni nahlížíme. Vločce tak můžeme ponechat dvoudimenzionální vlastnosti, tzn. nechat ji se neustále natáčet vůči pohledu kamery bez dalších úprav (tzv. billboarding).

Tvar vločky je opět definován v geometry shaderu, tentokrát jakožto pravidelný osmiúhelník. Vzhledem k velikosti a pohybu vločky se bude při běhu programu jevit jako kruh.

Pohyb vločky je definován složitěji, než u deště – zatímco u dešťových kapek měli odpor vzduchu, poryvy větru a další vnější činitelé zanedbatelný vliv na její trajektorii, u sněhové vločky by rovnoměrný pohyb všech vloček ve stejném směru vypadal nepřirozeně. Vločky proto byly rozděleny do deseti stejně velkých skupin, které každá měla nadefinovaný směrový vektor jiný, ale stále podobný vektorům skupin ostatních. Tímto rozdělením je zajištěno, že se vločky sněhu budou pohybovat zhruba ve stejném směru, avšak zdánlivě eraticky a nerovnoměrně.

Výslednou podobu sněhových vloček ve scéně je možné vidět na obrázku 4.3.

Obrázek 4.3: Sněhové vločky si zachovávají stejnou intenzitu bílé barvy po celé své ploše.

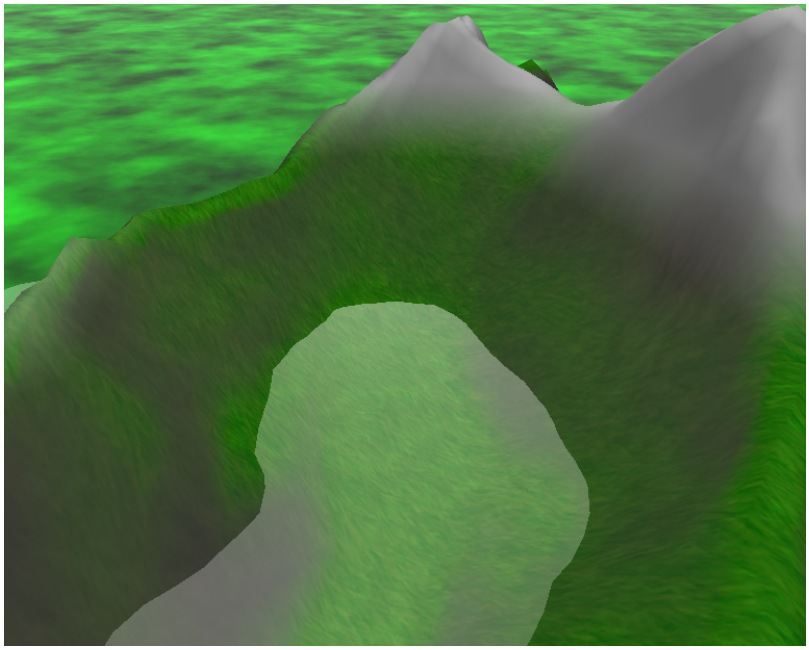
4.2 Terén

Terén je procedurálně vygenerován z výškové mapy, tzn. bitmapového obrázku ve stupních šedi. Barvy jednotlivých pixelů poté udávají výslednou výšku daného bodu, čím světlejší pixel tím vyšší bod. Po načtení takového obrázku vzniknou bodové souřadnice tvořící síť (anglicky *mesh*) terénu, ze kterých se pro každou jimi ohraničenou plochu (lze si představit jako oka rybářské sítě) vypočítají normály, které jsou posléze využity k implementaci osvětlení. Normála je počítána pro každý vertex a je normalizovanou sumou normálových vektorů spočítaných z ploch sousedících s vertexem pomocí vektorového součinu. Posledními souřadnicemi posílanými do shaderů jsou pak texturové koordináty.

K realizaci osvětlení je využito principů Phongova osvětlovacího modelu, avšak bez odlesků. Je tomu tak proto, že terén není lesklým objektem a přítomnost lesklé složky by na něm proto neměla smysl. Osvětlení terénu je tedy řešeno kombinací ambientního a difúzního světla. Difúzní světlo je pak počítáno právě ze získaných normál. Díky pohybujícímu se zdroji světla je možné pozorovat postupné prodlužování a prohlubování se stínů.

Poslední součástí tvorby terénu je procedurální přidělení textur. Do programu jsou nahrány dva bitmapové obrázky, jeden pro travnatou plochu a druhý pro skalnatou. V závislosti na úhlu mezi normálou každé plochy terénu a normálou roviny tvořené osami x a z je pak pro danou plochu přidělena jedna z textur, konkrétně skalnatá textura pro plochy strmější než šedesát stupňů a travnatá pro plochy méně strmé než třicet stupňů. Plochy mezi těmito hodnotami pak mají předěleny textury obě naráz v poměru závislém na strmosti. Podobným způsobem je pak řešena sněhová pokrývka na vyšších částech terénu, kde je podle dané výšky k texturám přidána bílá barva. Výsledný efekt je si možné prohlédnout na obrázku 4.4.

Implementační součástí terénu je i mlžný opar, ten je však dále rozveden v sekci o mlze.



Obrázek 4.4: Čím strmější je plocha terénu, tím více je v ní patrná textura skály.

4.3 Mlha

Implementace mlhy je rozdělena do dvou částí – hranice mlhy a mlžný opar. Hranicí mlhy je zde míněna její boční a horní hranice, pod níž začíná mlžný opar a která je implementována v samostatných shaderech. Mlžným oparem je zde míněna postupná ztráta viditelnosti v závislosti na vzdálenosti od pozorovaného bodu. Ta je implementována v rámci terénových shaderů.

4.3.1 Hranice mlhy

Pro hranici mlhy jsou využité souřadnice terénu vypočítané v rámci hlavního programu (bližší popis viz sekce 4.2). Tyto body jsou přivedeny do vertex shaderu mlhy, kde jsou pomocí šumových funkcí deformovány tak, aby jejich souhrnný tvar byl odlišný od tvaru terénu, avšak stále podobný. Takto jsou vytvořeny dvě „vrstvy“ mlhy (každá deformována trochu jinak) které jsou posunuty zhruba do poloviční výše terénu a poslány první po sinové, druhá po kosinové funkci v čase. Z bodů obou vrstev, kdy je vždy vybrán ten vyšší,

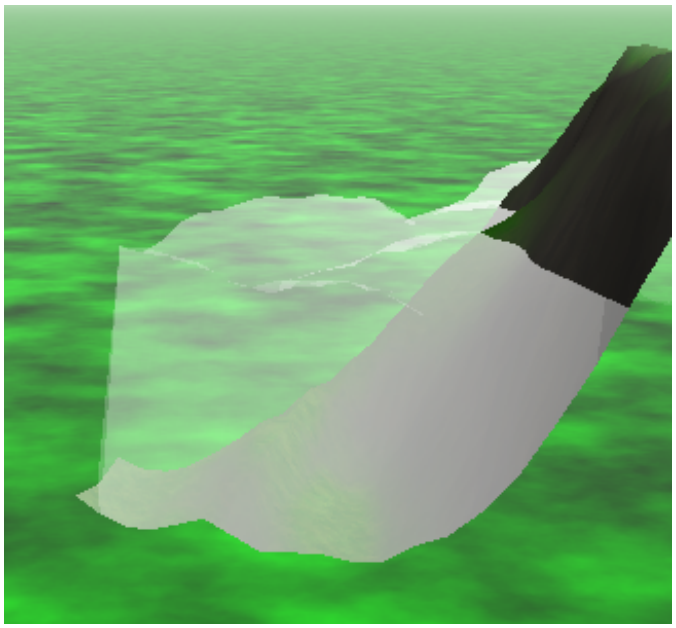
je vytvořena vrstva finální, která je posléze i vykreslena. Díky neustále se měnícím polohám bodů se finální vrstva různě vlní a dodává tak mlze dynamický vzhled.

Barva hranice mlhy se mění v závislosti na denní době a její body se stávají průhlednějšími pokud se k nim přiblíží kamera. Výpočet průhlednosti je dán matematickou rovnicí 4.1.

$$p = \frac{1 - (k - v)}{k - s} \tag{4.1}$$

V rovnici 4.1 značí p výslednou průhlednost, k vzdálenost, za kterou nebude průhlednost žádná, v vzdálenost kamery od pozorovaného bodu a s nejmenší možnou vzdálenost, v jaké je mlhu možné pozorovat. Pokud je tedy bod ve vzdálenosti menší než s, pak není viditelný.

Výsledný vzhled hranice mlhy ilustruje obrázek 4.5.



Obrázek 4.5: Díky generování hranice mlhy i na jejích stranách bude při pohledu z mlhy zamlžené i pozadí.

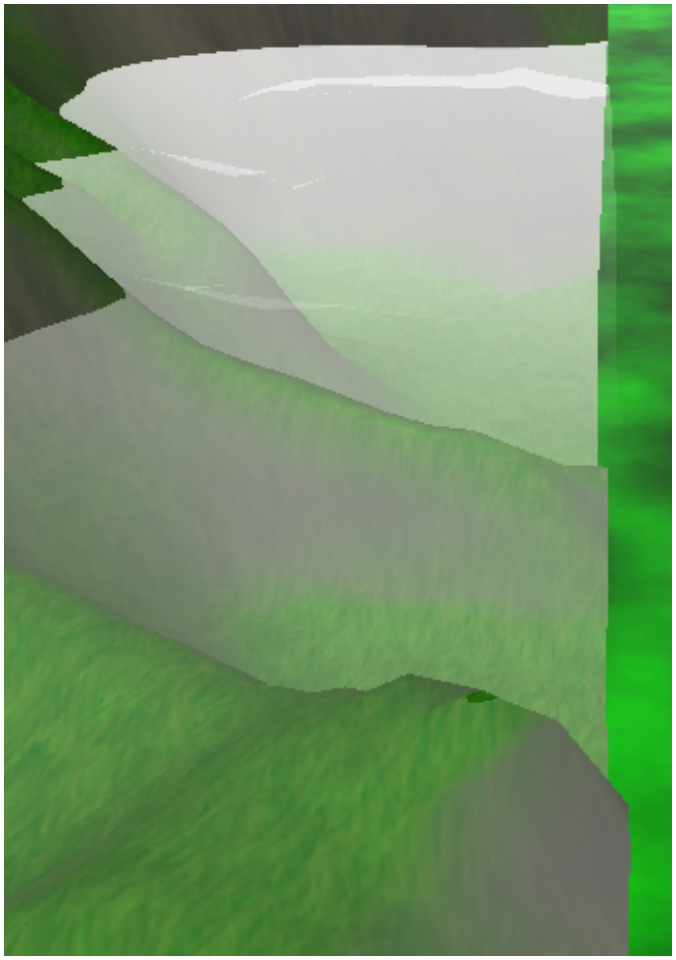
4.3.2 Mlžný opar

Mlžný opar také vypočítává hranici mlhy, avšak nevykresluje ji. Místo toho ji používá jakožto svou limitu. Program pro každý bod terénu zjišťuje, zda-li je pod či nad touto hranicí a pokud je nad ní, hodnotí vzdálenost bodu od kamery jako nulovou. Pokud je pod ní pak je vzdálenost vypočítána pomocí Pythagorovy věty a dosazena do rovnice 4.2.

$$d = \frac{k - v}{k - s} \tag{4.2}$$

Písmeno d značí v rovnici 4.2 mlhový faktor pohybující se v intervalu <0,1>. Písmena k, v a s značí to samé, co v předchozí podkapitole, tzn. vzdálenost za kterou bude mít mlha stoprocentní neprůhlednost, vzdálenost kamery od bodu a vzdálenost od které začne být opar patrný.

Do výsledné barvy terénu je poté přimíchána barva mlžného oparu, která závisí na denní době, a to v poměru udaném mlhovým faktorem. Výsledný vzhled je ukázán na obrázku 4.6.



Obrázek 4.6: Původní barva terénu ustupuje ve prospěch barvy oparu tím více, čím dále od kamery je pozorovaný bod.

4.4 Pozadí

Obloha a krajina pod ní (nikoli modelovaný terén) jsou řešeny tzv. animovaným skyboxem. Jedná se o umístění obdélníkového objektu na zadní průmětnu ve dvoudimenzionálním prostoru, který zabírá celý zorný úhel kamery bez toho, že by jej někdy opustil (tzn. pohybuje se spolu s kamerou ve stálé vzdálenosti od ní). Veškerý pohyb na něm je tedy ve skutečnosti pouze změnou barvy určitých jeho částí, nikoli pohybem objektů.

Barva jednotlivých částí pozadí je přidávána postupně, po vrstvách, kdy je barva nové vrstvy přimíchána k barvě staré, případně ji přemaluje úplně. Princip této techniky spočívá ve využívání průhlednosti bodů, kdy sice každá vrstva přiděluje novou barvu každému bodu, avšak u bodů, které měnit nechce, učiní barvu zcela průhlednou a tedy neschopnou ovlivnit barvy nižších vrstev. Poměr barev v daném bodě je pak dán jejich lineární interpolací.

Základní vrstvou nanesenou před všemi ostatními je černá barva. Následující podkapitoly jsou logicky seřazeny podle pořadí aplikace jednotlivých vrstev v programu.

4.4.1 Hvězdy

Hvězdy jsou generovány po celé ploše pozadí pomocí jednoduchého generátoru pseudonáhodných čísel. Program však učiní neprůhlednými pouze body s velmi vysokou vygenerovanou hodnotou, což zajistí, že většina pozadí zůstane nadále černá.

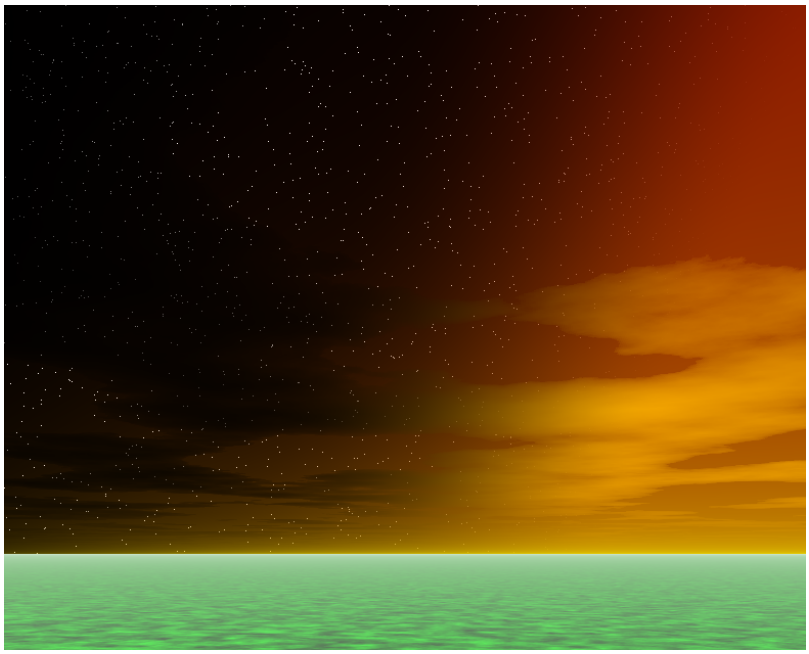
4.4.2 Nebe

Pojmem nebe je v této sekci míněna pouze jeho denní část, nikoli část noční. Zatímco tato vrstva totiž významně mění vizuální stránku denní části oblohy, body v noční části ponechává zcela průhledné a ovlivněné tedy pouze základní černou vrstvou a vrstvou hvězd.

Barva nebe je dána interpolací mezi barvou horizontu a barvou oblohy, jejichž poměr určuje odmocnina ypsilonové složky směrového vektoru do daného bodu z počátku souřadnicové soustavy ve world space. Jinými slovy čím blíže je bod rovině tvořené osami x a z, tím výraznější bude vliv barvy horizontu. Poměr barev je dán jejich lineární interpolací.

Obě barvy jsou dále ovlivněny přechody mezi denní a noční částí vážícími se na slunce, které bodům dávají tím vyšší průhlednost, čím větší je úhel mezi jejich směrovým vektorem a vektorem slunce (viz podsekce 4.4.3). Tím je docíleno rozdělení nebe na denní a noční část a zabránění horizontu v přesahování do noční oblohy.

Konkrétní barvy horizontu a nebe jsou závislé na denní době a pohybují se v odstínech červené, žluté, modré a černé. Výsledný efekt je si možné prohlédnout na obrázku 4.7.



Obrázek 4.7: Barva nebe i horizontu postupně slábnou, dokud nedosáhnou stoprocentní průhlednosti.

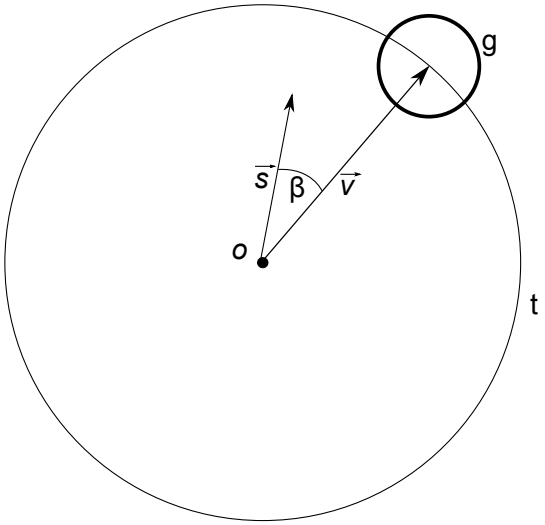
4.4.3 Slunce

Slunce putuje ve scéně po kruhové trajektorii implementované vektorem směřujícím od počátku souřadnicové soustavy. Podoba vektoru je znázorněna v rovnici 4.3.

$$vektor = (0, -\cos\alpha, \sin\alpha) \tag{4.3}$$

V rovnici 4.3 je α hodnota ovlivněná časem, což slunci umožňuje pohyb. Pro každý bod scény je pak vypočítán skalární součin vektoru z počátku souřadnicové soustavy k němu a vektoru slunce. Pokud je součin dostatečně velký (a úhel mezi vektory je tedy malý), bude příslušným bodům dána nulová průhlednost. Bodům na samém okraji slunce je pak dána sestupná průhlednost simulující sluneční paprsky. Zbylé body scény dostávají průhlednost stoprocentní. Celý princip zobrazuje schéma 4.8.

Barva vrstvy závisí na denní době a pohybuje se v odstínech žluti.



Obrázek 4.8: Slunce se pohybuje po trajektorii t, přičemž jeho pozici definuje vektor $\vec{v}$ (viz 4.3) s počátkem ve středu souřadnicové soustavy o. Vektor $\vec{s}$ je vektorem z o do právě vykreslovaného bodu pozadí. Úhel svíraný vektory $\vec{s}$ a $\vec{v}$ je nazvaný β a rozhoduje o průhlednosti bodů v rámci této vrstvy. Pomyslnou hranici slunce pak značí g – body uvnitř této hranice budou neprůhledné, body vně průhledné a body ležící na hranici samotné budou průhledné částečně.

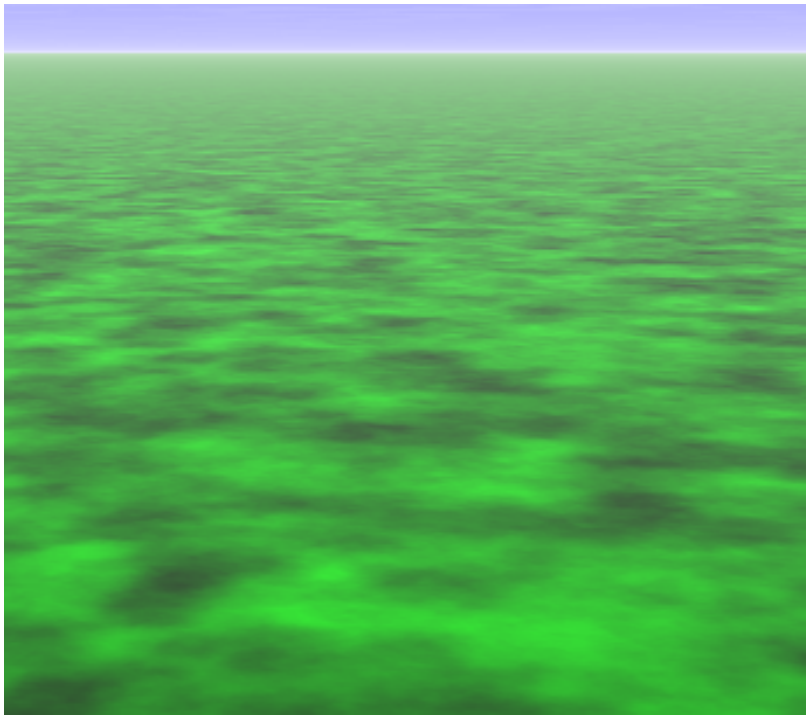
4.4.4 Krajina

Pojmem krajina zde není myšlen modelovaný terén, ale oblast na pozadí, kterou můžeme pozorovat při pohledu kamerou na horizont, případně při pohledů dolů když se kamera nenachází nad terénem. Jednoduchá implementace krajiny je v programu přítomna pro lepší orientaci uživatele v prostoru a pro lepší vizuální atraktivitu.

Pozice krajiny na pozadí je definována velmi jednoduše – pokud se vykreslovaný bod nachází v dolní polovině pomyslné koule obklopující celou scénu (pokud je ypsilonová složka vektoru směrem od počátku souřadnicové soustavy k danému bodu ve world space nižší, než nula), bude neprůhledný, jinak mu bude nastavena průhlednost stoprocentní.

Barva krajiny je dána interpolací mezi bílou barvou a zelenou barvou, jejíž odstíny jsou určeny pomocí Perlinova šumu. Poměr mezi nimi určuje odmocnina z ypsilonové složky směrového vektoru do daného bodu (tzn. čím blíže bude bod hranici mezi nebem a zemí, tím nižší tato hodnota bude), což simuluje horizont. Prakticky identický postup byl již využit v sekci 4.4.2.

Výsledný efekt je pak možné vidět na obrázku 4.9.



Obrázek 4.9: Světlejší body znázorňují kopce a tmavé údolí, což dodává krajině lepší vzhled.

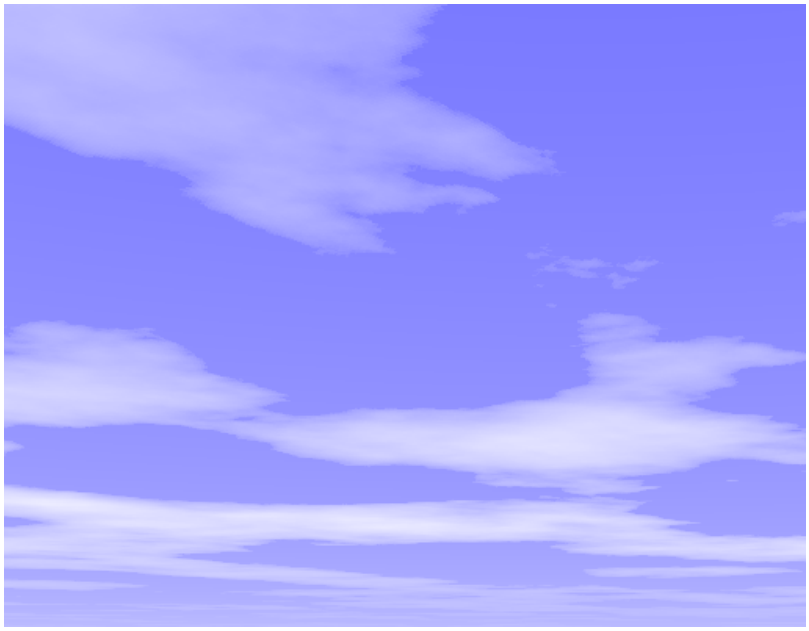
4.4.5 Mraky

Mraky jsou definovány dvěma funkcemi, první z nichž popisuje jejich hustotu a druhá jejich barvu.

Hustota mraku v daném bodě je udána Perlinovým šumem jehož vstupní parametry jsou ovlivněny časem, a tedy zajišťují, že se mraky budou během běhu programu pohybovat. Hodnota hustoty mraku je poté využita jakožto hodnota průhlednosti.

Barva mraků je dána barvou slunce (respektive denní dobou) vynásobenou postupným přechodem mezi denní a noční oblohou, který zajišťuje, že se mraky zbarví do černa pokud začnou zasahovat do noční části.

Ukázku mraků ve výsledné scéně je možné si prohlédnout na obrázku 4.10.



Obrázek 4.10: Mraky jsou generovány pomocí Perlinova šumu, který zajistí, že jejich body nebudou distribuovány náhodně, ale budou se skupovat do shluků.

Kapitola 5

Závěr

Cílem této bakalářské práce bylo vytvořit aplikaci, která by vizualizovala krajinu a různé atmosférické jevy s důrazem na estetický vzhled za použití procedurální generace. Tohoto cíle bylo dosaženo – program dokáže z výškové mapy vygenerovat terén a následně ho otexturovat, zobrazuje čtyři atmosférické jevy (déšť, sníh, soumrak, mlha) a v implementaci výrazně používá techniky procedurální generace. Estetická hodnota výsledné scény je, dle mého názoru, vysoká a to zejména díky animovanému skyboxu s pohybujícími se mraky a měnící se denní dobou, dynamicky přidělovaným texturám a pohybu slunce, jež zajišťuje postupné změny stínů terénu.

Díky vysoké modularitě programu je možné aplikaci jednoduše rozšířit o další atmosférické jevy, či jevy existující dále vylepšit. Jako příklad rozšíření se nabízí jevy jako jsou blesky nebo kroupy.

Literatura

[1] *Khronos Group – Transform Feedback*. [Online; navštíveno 10.05.2017].
URL `https://www.khronos.org/opengl/wiki/Transform_Feedback`

[2] *Microsoft – Billboarding*. [Online; navštíveno 10.05.2017].
URL `https://msdn.microsoft.com/en-us/library/windows/desktop/bb324472(v=vs.85).aspx`

[3] *Techmania Science Center – Atmosférické jevy*. [Online; navštíveno 10.05.2017].
URL `http://edu.techmania.cz/cs/encyklopedie/fyzika/meteorologie/atmosfericke-jevy`

[4] Bajaj, C.: Illumination I: The Phong Illumination Model. USA, 2013, slajdy z přednášky.
URL
`http://www.cs.utexas.edu/~bajaj/graphics2012/cs354/lectures/lect14.pdf`

[5] Byška, J.: *Perlinův šum a jeho aplikace*. Bakalářská práce, Masarykova Univerzita. Fakulta informatiky, 2009.

[6] Černý, M.: *Generování terénu s využitím voxelové struktury*. Diplomová práce, Masarykova Univerzita. Fakulta informatiky, 2015.

[7] Gerledan, A.: Bui Tuong Phong's Lighting, University of Utah, 1973, but with shaders. Irsko, 2014, slajdy z přednášky.
URL `http://antongerdelan.net/teaching/cs4052/05_lighting_lecture.pdf`

[8] Jurík, A.: *Fractal Terrain Generation*. Bakalářská práce, Masarykova Univerzita. Fakulta informatiky, 2016.

[9] Peringer, P.: Modelování a simulace. Brno, 2016, slajdy z přednášky.
URL `https://wis.fit.vutbr.cz/FIT/st/course-files-st.php?file=%2Fcourse%2FIMS-IT%2Flectures%2FIMS.pdf&cid=11453`