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

SIMULACE SNĚHOVÝCH A LEDOVÝCH STRUKTUR

SIMULATION OF SNOW AND ICE FEATURES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ONDŘEJ ČECH**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing. MARTIN ČADÍK, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Čech Ondřej**
Program: Informační technologie
Název: **Simulace sněhových a ledových struktur**
Simulation of Snow and Ice Features
Kategorie: Počítačová grafika

Zadání:

1. Seznamte se s existující literaturou o simulaci a vizualizaci kajícníků, resp. séraků.
2. Navrhněte a implementujte systém pro vizualizaci kajícníků, včetně simulace v časové dimenzi.
3. S navrženým systémem experimentujte, vyhodnoťte dosažené výsledky a diskutujte možnosti budoucího vývoje.
4. Dosažené výsledky prezentujte formou videa a plakátu, příp. článku.

Literatura:

- Dle pokynů vedoucího.
- http://en.wikipedia.org/wiki/Penitentes
- Bergeron V, Berger C, Betterton MD. Controlled irradiative formation of penitentes. Phys Rev Lett. 2006 Mar 10;96(9):098502.

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

Vedoucí práce: **Čadík Martin, doc. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 6. listopadu 2019

Abstrakt

Tato práce se zabývá simulací vzniku ledovcových struktur, známých jako Kajícníci. Simulace je založena na matematickém modelu formování těchto struktur vytvořeném M. D. Betterton. Implementace je realizována v jazyce C++, výstupem je rastrový obrázek obsahující výškovou mapu ledové vrstvy, kterou lze vykreslit jako 3D objekt pomocí programu Blender.

Abstract

This project focuses on simulation of formation of glaciers known as Penitentes. It is based on mathematical model of formation of these structures by M. D. Betterton. Implementation is done in C++ programming language, and the result is exported as bitmap image containing heightmap, which can be rendered using Blender.

Klíčová slova

Kajícníci, Penitentes, Ledovce, C++, Simulace

Keywords

Penitentes, Glaciers, C++, Simulation

Citace

ČECH, Ondřej. *Simulace sněhových a ledových struktur*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Martin Čadík, Ph.D.

Simulace sněhových a ledových struktur

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. Ing. Martina Čadíka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Ondřej Čech
28. května 2020

Poděkování

Chtěl bych zde poděkovat svému vedoucímu práce za vedení, pomoc a rady v průběhu práce na tomto projektu.

Obsah

Kapitola 1

Úvod

Cílem práce je vytvořit simulaci toho, jak vznikají ledovcové struktury známé jako Kajícníci neboli Penitentes na základě známých informací o tomto jevu. Pro řešení práce je tím pádem potřeba nastudovat dosavadní výzkum a teorie ohledně vzniku a chování Kajícníků a rovněž vybrat vhodnou vizualizační metodu na zobrazení výsledku.

Kajícníci se obvykle nacházejí ve vysokohorském prostředí a mají podobu vysokých plochých ledových štítů natočených nejmenší plochou k západu a východu. Problém je, že tyto struktury nejsou stále dokonale prozkoumány a jejich přesný způsob vzniku tedy není znám. Jejich první popis pochází z třicátých let 19. století, kdy je poprvé uviděl a popsal Charles Darwin. Během času přišli různí vědci s teoriemi jejich růstu, bylo provedeno mnoho měření a pokusů. V současné době obecně panuje přesvědčení, že za vznik může sluneční záření, které ve spolupráci s dalšími podnebními faktory může za úbytek sněhu v okolí, zatímco tato struktura přežívá. V nedávné době se povedl experiment vytvoření kajícníků uměle v laboratoři na základě této teorie. Význam znalosti vzniku kajícníků spočívá v jejich schopnosti zpomalit tání ledovce, protože tvoří větší povrch, který se prouděním studeného vzduchu ochlazuje, a rovněž rozdíly mezi teplotami uvnitř prohlubní a na vrcholech mohou vést k tomu, že část odpařené vody v prohlubních opět zkondenzuje na vrcholech a voda tak zůstane v ledovci [8].

Tato práce je rozdělena na pět hlavních kapitol.

V první části je popsána historie výzkumu jevu, teorie vzniku těchto struktur a matematický model tohoto vniku, který vypracovala Meredtith D. Betterton. Poté se tato kapitola věnuje způsobům zjištění pozice Slunce na obloze a pojmům, které s touto problematikou souvisejí.

Následující kapitola se věnuje připravenému návrhu programu. Zde jsou v první části popsány technologie, které budou během realizace použity. Druhá část kapitoly popisuje samotný způsob, jak bude program řešen.

Další kapitola popisuje implementaci daného modelu. Zde jsou popsány jednotlivé soubory programu a funkce v nich uvedené. Obsahem předposlední kapitoly jsou experimentování na modelu, výsledky a jejich vyhodnocení, poslední je závěr, který shrnuje dosavadní dosažené výsledky, dokončenou práci a určuje cíle pro následující práci na projektu.

Kapitola 2

Teoretické podklady pro simulaci

Tato kapitola se věnuje teoretickým informacím, které slouží jako podklady pro tuto práci. Je věnována zejména útvarům známým jako Kajícníci, historii jejich výzkumu a výsledkům těchto zkoumání, včetně matematického popisu teorie jejich vzniku. Také je zde část věnující se pozici Slunce na obloze a souvislosti této pozice se vznikem Kajícníků.

2.1 Kajícníci

Útvary nazývané Kajícníci (španělsky Penitentes) jsou struktury z ledu nebo sněhu, které mají podobu tenkých zašpičatělých štítů dosahujících výšky až 5 metrů. Tyto útvary vznikají ve vysoké nadmořské výšce v období ablace, což je časový interval, kdy ledovec ztratí více objemu, než získá. Běžně se vyskytují v Andách mezi Chile a Argentinou v nadmořské výšce 4000 až 5200 metrů nad mořem, také se objevují v Himalájích, pohoří Sierra Nevada, na Kilimandžáru a jiných horách s vyhovujícím klimatem, rovněž byly takovéto útvary nalezeny na Plutu a mohly by se nalézat v rovníkových oblastech Jupiterova měsíce Europa [3]. Klima, které vyhovuje růstu Kajícníků lze popsat jako suché, studené, slunečné a stabilní. Ve výškách nad 5200 metrů zůstává sníh příliš prašný, aby se v něm mohli kajícníci vytvořit [13].

Kajícníci tvoří skupiny štítů orientovaných obecně ve směru z východu na západ se zdmi obrácenými k severu a jihu, ačkoli v konkrétních případech mohou vznikat struktury s mírně odlišnou orientací. Příčinou toho jsou další faktory v daném místě, například tvary terénu pod sněhovou pokrývkou nebo tvorba závějí. [5]. Jejich vrcholy se orientují k zenitálnímu úhlu slunce během poledne, kdy je jeho záření nejvyšší. Název kajícníků pochází od jejich vzhledu, který připomíná účastníky procesí během Velikonočních svátků ve Španělsky mluvících zemích. Ti jsou oblečeni v dlouhých bílých hábitech a vysokých špičatých pokrývkách hlavy, nazývaných *capirote*.

2.2 Historie výzkumu Kajícníků

Poprvé tyto útvary popsal britský přírodovědec Charles Darwin, který se s nimi setkal při cestě ze Santiago de Chile do Argentinského města Mendoza v březnu 1835. Podle místních obyvatel za jejich tvorbou stojí silné větry [10]. Ohledně jejich vzniku vzniklo časem několik dalších teorií, například Argentinský geolog Luciano Roque Catalano došel roku 1926 k závěru, že vznikají působením intenzivního ultrafialového záření a suchého větru. Jejich vrcholy podle něj ve dne tají a v noci namrzají, přičemž vzniká elektrické

Obrázek 2.1: Pole kajícníků v Argentině, poblíž hory Aconcagua [1]

pole, díky čemuž se krystalky rovnají do pravého úhlu k magnetickému poli Země. Louis Lliboutry roku 1954 zjistil, že za vznikem kajícníků stojí sluneční záření ve chvíli, kdy je rosný bod pod bodem mrazu, což vede k ablaci ledu sublimací [13].

Roku 2006 Vance Bergeron, Charles Berger, a M. D. Betterton poprvé vytvořili kajícníky v laboratoři [6]. Ve velkém mrazáku zakrytém plexisklem svítili na kus ledu, přičemž vzduch vcházející dovnitř byl chlazen tekutým dusíkem. Jako zdroj záření byl použit světlomet, světelný zdroj umístěný v zrcadlovém odražeči, který směřuje všechny paprsky záření stejným směrem. Během několika hodin se na ledu objevily 1–5 cm vysoké ostny. Podobné experimenty prováděl už ve třicátých letech Německý geolog Carl Troll, který za suché chladné noci svítil na čerstvý sníh žárovkou [13].

2.3 Tvorba

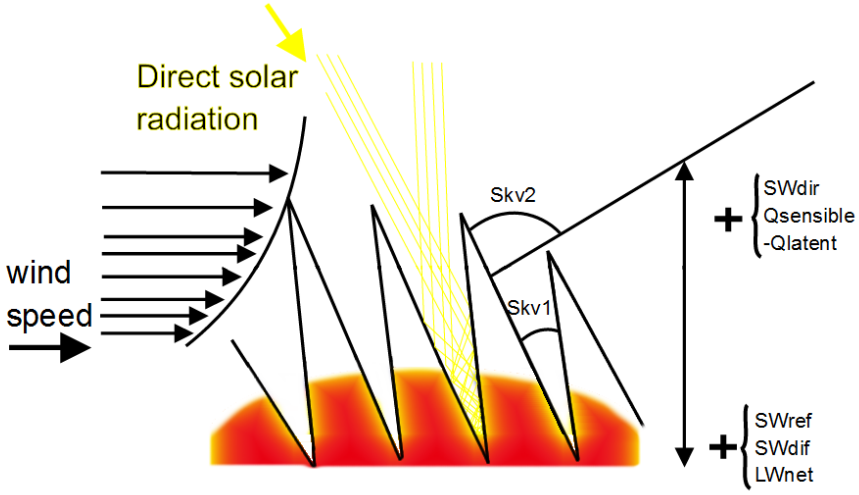
Tato kapitola vychází z poznatků uvedených v [7] a [13]. Proces, který vede ke vzniku kajícníků se nazývá diferenciální ablace. Při ní mění některé části sněhu své skupenství rychleji než jiné. Na počátku je silná vrstva sněhu, která není dokonale rovná, jsou v ní propadliny. Na tento sníh dopadá sluneční záření, které je zachycováno vícenásobnými odrazy mezi zdmi těchto sníženin více, než jinde. Tento jev se nazývá pozitivní zpětná vazba. Jak již bylo zmíněno, klíčový faktor pro tvorbu kajícníků je rosný bod pod bodem mrazu. Takto nevzniká voda v tekutém skupenství, ale sníh sublimuje, což je energeticky náročnější, a výdej této energie vede k ochlazování ledovce. Propadliny se vlivem větší přijaté energie prohlubují, stávají se téměř černým tělesem, což znamená, že pohlcuje téměř všechno světlo, co na něj dopadá. Zároveň uvnitř roste hodnota rosného bodu, která se může dostat až nad bod mrazu, a sníh uvnitř začne tát. Oproti tomu vrcholy kajícníků stále jen sublimují a jejich

Obrázek 2.2: Účastníci procesí v Granadě [11]

strmé stěny zachytí jen velmi málo slunečního záření. Průběh vzniku pole kajícníků zachycují časosběrné snímky, která pořídila glaciologická skupina CEAZA (Centro de Estudios Avanzados en Zonas Áridas) v Chile od konce listopadu 2010 do června 2011, které lze najít na YouTube[1]. Podobným způsobem vznikají další sněhové útvary zvané suncups, což jsou skupiny miskovitých prohlubní těsně vedle sebe. V případě znečištěného povrchu vznikají takzvané *dirt cones*. To je častý jev například na Islandu, případně po výbuchu silné sopky. Prach se usadí v prohlubni a tak chrání sníh pod sebou před zahříváním, zatímco sníh kolem roztaje. Sníh v blízkém okolí prohlubně tak pod ochrannou vrstvou prachu vystoupí nad povrch [14].

Značnou roli v procesu hraje i vítr. Podle experimentů provedených Takahashim a spolupracovníky rostou tyto struktury nejrychleji, pokud je rychlost a teplota vzduchu co nejvyšší [15]. Proudění vzduchu ale vyrovnává teplotu i vlhkost vzduchu po celém poli kajícníků. Dalším faktorem ovlivňujícím růst kajícníků je znečištění. Tmavé nečistoty pohlcují sluneční záření, ale velmi záleží na síle vrstvy těchto nečistot. Tenkou vrstvou část záření projde, ale už se neodrazí zpět, což ablaci urychluje, naopak moc silná vrstva znečištění ablaci zpomaluje, protože záření zůstává zachyceno v nečistotách a na sníh se už nedostane. V rámci laboratorního pokusu popsaného dříve také jeho autoři zkusili posypat led tenkou vrstvou sazí. Na vrcholcích tvořících se kajícníků se saze koncentrovaly a tvořily tak stále silnější vrstvu, která je začala chránit před zářením. V prohlubních se zvětšovala plocha a tím pádem se koncentrace sazí snižovala. Toto vedlo k rychlejší tvorbě útvarů, jelikož vrcholy tály ještě pomaleji, než kdyby byly z čistého sněhu, zatímco údolí se prohlubovala zhruba stále stejně rychle.

[1] https://www.youtube.com/watch?v=HjlkrVxBtIA



Obrázek 2.3: Schéma zobrazující princip růstu kajícníků. Sluneční záření je díky vícenásobným odrazům od stěn intenzivnější v prohlubních. S rostoucí výškou klesá celkové dlouhovlnné záření díky větší viditelnosti oblohy. Celkové turbulentní toky jsou na vrcholcích, kde převládá sublimace, spíše negativní, což vede k ochlazování. Schéma bylo převzato z [9] a upraveno.

2.4 Matematický model vzniku Kajícníků

Meredith D. Betterton vytvořila ve své práci [7] matematický model tvorby kajícníků. Tento model uvažuje sluneční záření jako primární zdroj tepla, který vede k ablaci. Je ale poněkud zjednodušený v některých aspektech. Latentní teplo, tedy energii, kterou je nutné dodat ke změně skupenství, uvažuje jako konstantní, zahrnuje izotropní odrazy pouze prvního řádu a zachycuje pouze jednodimenzionální přístup s předpokladem invariantního přístupu vůči ostatním směrům. Tento model se skládá z dvou částí: výpočtu sněhové ablace a odrazivosti světla.

2.4.1 Ablace

Teplo, které dopadá na povrch, vede k ablaci. Výška povrchu h se snižuje, jak led taje nebo sublimuje. Předpokládá se, že odtátý led se vsákne nebo vypaří, tedy nedochází k toku vody po povrchu a jejímu opětovnému zmrznutí. Povrch v okolí bodu dokáže absorbovat množství energie v závislosti na ploše okolí bodu. Tato energie pro bod x bude označena jako $P(x)$. Latentní teplo potřebné k ablaci objemové jednotky ledu je L. Kombinace tohoto ve spojení s efektivním difúzním vyhlazovacím termem tvoří rovnici vývoje povrchu:

$$\frac{\delta h}{\delta t} = \frac{P(x)}{L} + D\frac{\delta^2 h}{\delta x^2}, \tag{2.1}$$

kde D je difúzní konstanta a L je latentní teplo.

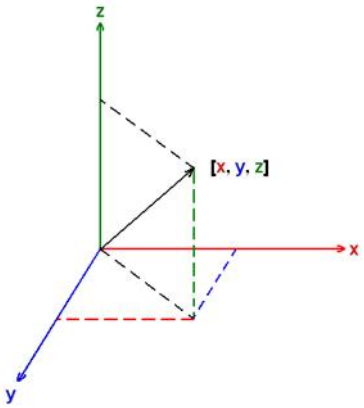
Jak již bylo zmíněno, L je uvažováno konstantní, což nastává, pokud teplota povrchu a vlhkost vzduchu jsou konstantní. Pro vyvinuté kajícníky toto nemusí úplně platit, pro malé struktury s malými úhly by ale konstantní L mělo být dostatečně přesné. Latentní teplo závisí na hustotě. Zatímco čerstvě napadaný sníh má hustotu mezi $0,05$ a $0,2\,g\,cm^{-3}$,

zatímco sníh, který přežil již jednu sezónu tání, má hustotu v rozmezí od $0,4$ do $0,8\,g\,cm^{-3}$. Pro tento model byla vybrána průměrná hodnota $0,3\,g\,cm^{-3}$. To znamená, že latentní teplo na danou objemovou jednotku je rovno $L = 10^9\,erg\,cm^{-3}$ a rychlost tání $I/L = 10^{-3} cm\,s^{-1}$. *Erg* je jednotka energie v soustavě CGS, kde je její rozměr $g\,cm^2 s^{-2}$. Odpovídá tak hodnotě $10^{-7}\,J$. Pro sublimaci sněhu je potřebné latentní teplo sedminásobně vyšší. Druhý term v dané rovnici je difúzní term s difúzní konstantou D. Bez vyhlazovacího termu může model tvořit libovolně malé struktury, což neodpovídá realitě. Proto je třeba mít nějakou velikost nejmenší vytvořitelné struktury. Přirozeným omezením je například délka zániku slunečního světla, která definuje tloušťku sněhové vrstvy, ve které probíhá rozptyl světla. Jako D byla vybrána hodnota $2,5 \times 10^{-5} cm^{-2} s^{-1}$ tak, aby nejméně stabilní byla vlnová délka 2 cm.

Body na povrchu vzdálené od sebe méně, než je tato délka, nejsou opticky nezávislé, a proto tyto body odtávají stejnou rychlostí. Délka zániku je závislá na hustotě a struktuře zrn sněhu. Typická délka zániku pro sníh o poloměru 1 mm je v řádu 1 cm.

2.4.2 Odraz světla

Sluneční světlo dopadá na povrch přímo seshora, jinak řečeno opačným směrem k ose Z prostorové kartézské soustavy souřadnic znázorněné na 2.4 a přenáší energii. Tato energie je konstantní vůči ploše kolmé na směr paprsků. Tuto vlastnost popisuje Solární konstanta I, která udává intenzitu solárního záření v nejvyšším bodě atmosféry. $I = 1,4 \times 10^6\,erg\,cm^{-2} s^{-1}$. Proto je jako typická hodnota I v jasných slunečných podmínkách bráno $10^6\,erg\,cm^{-2} s^{-1}$. Parametr α reprezentující odrazivost povrchu se nazývá *albedo*. Ten označuje, jak velká část světla se odrazí. Pro práci je uvažován sníh označovaný jako *Old snow*. Jedná se o uležený sníh, který už nemá rozpoznatelnou strukturu původních jednotlivých krystalků. Často se používá i označení *Firn*, které popisuje zakulacený, dobře propojený sníh, který přežil minimálně jednu ablační sezónu, ale ještě není ledovcem, protože právě z takového sněhu se následně ledovce tvoří. Firn má hustotu větší než $550\,kgm^3$. Pro takový sníh je pak albedo $\alpha = 0,5$.



Obrázek 2.4: Kartézská soustava souřadnic v prostoru. Převzato z [2].

Hlavní rozdíl v odrazu světla od sněhu a odrazu světla od zrcadla je ten, že jelikož se nejedná o hladký povrch, světlo se odráží různými směry a proto se sníh jeví bílý. Za předpokladu, že povrch odráží světlo izotropně, dochází k rovnoměrnému rozdělení energie

v prostorovém úhlu o velikosti 2π ve směru od povrchu. Odraz od povrchu je zde zjednodušení, jelikož k odrazu dochází ve vrstvě zhruba 1 cm pod povrchem.

Při použití těchto vlastností je celkové množství světla Φ odraženého z intervalu okolo bodu x_1 do intervalu mezi body x a dx:

$$\Phi = \int \frac{\alpha I}{\pi} d\theta \, dx_1, \tag{2.2}$$

kde $d\theta$ je úhel s vrcholem v bodě x_1 a rameny do bodů x a $x + dx$, $d\theta$ lze vyjádřit jako:

$$d\theta = \frac{dl}{p} = \frac{|\mathbf{p} \times d\mathbf{s}|}{p} = \frac{\Delta h - \Delta x h'(x)}{\Delta h^2 + \Delta x^2}, \tag{2.3}$$

kde je definován vektor $\mathbf{p}$, který směřuje z bodu x_1 do bodu x. Platí:

$$p = |\mathbf{p}| = \sqrt{\Delta h^2 + \Delta x^2}. \tag{2.4}$$

Dále dl je normála na $\mathbf{p}$ a ds je vektor představující tangentu k povrchu

$$d\mathbf{s} = dx(1, h'). \tag{2.5}$$

Rovněž platí následující vztahy:

$$\Delta x = x_1 - x \tag{2.6}$$

$$\Delta h = h(x_1) - h(x) \tag{2.7}$$

$$h' = \frac{\delta h}{\delta x}. \tag{2.8}$$

Pro zjištění celkové energie odražené do bodu x je třeba připočítat intenzitu ze všech bodů z okolí bodu x_1

$$P_r(x) = \frac{\alpha I}{\pi} \int \frac{dx_1(\Delta h - h'(x)\Delta x)}{\Delta h^2 + \Delta x^2}. \tag{2.9}$$

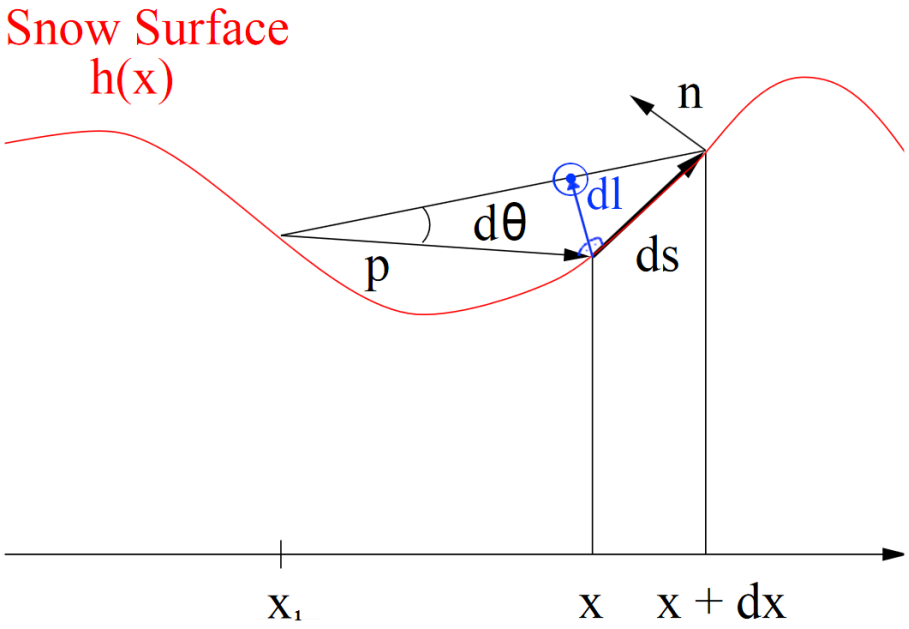
Integrovaná funkce je propagátor intenzity, popisuje, jak je intenzita přenášena z jednoho bodu do druhého. $P_r(x)$ je v tomto případě intenzita k jedinému odrazu. Pro zakomponování vícenásobných odrazů by bylo třeba rovnici upravit následovně

$$P(x) = (1 - \alpha)I + \frac{\alpha}{\pi} \int \frac{dx_1 P(x_1)(\Delta h - h'(x)\Delta x)}{\Delta h^2 + \Delta x^2}. \tag{2.10}$$

Tento model ale předpokládá pouze odrazy prvního řádu, proto bude jako výchozí nadále uvažován vzorec 2.9. Ten ještě ale stále není kompletní, protože nepočítá s omezením viditelnosti. Světlo se nemůže šířit z bodu x_1 do x, pokud v přímé cestě stojí překážka v podobě jiné části povrchu. Toto je nelineární omezení, které se těžko řeší analyticky, ale je řešitelné s využitím numerických metod. Toto omezení se obvykle vyznačuje schématicky pomocí textu „line of sight“ (přímá viditelnost) u symbolu integrálu

$$P_r(x) = \frac{\alpha I}{\pi} \int_{line\ of\ sight} \frac{dx_1(\Delta h - h'(x)\Delta x)}{\Delta h^2 + \Delta x^2}. \tag{2.11}$$

Lze zde zmínit jednoduchou, avšak obecně zcela nedostatečnou podmínku omezení viditelnosti dostačující v případě lokální analýzy jednoduché prohlubně. Body x_1 a x jsou ve vzájemné linii pohledu, pokud platí, že výsledek skalárního součinu vektoru $\mathbf{n}$ kolmého k povrchu v bodě x a vektoru $\mathbf{p}$ směřujícího z bodu x_1 do x (viz. schéma 2.5) je menší, než 0, tedy $-n \cdot p = \Delta h - \Delta x h'(x) > 0$. Toto omezení ale ignoruje objekty nacházející se mezi danými body. I v případě, že je podmínka splněna, nemusí ve skutečnosti k odrazu z bodu x_1 do x dojít, protože mezi nimi je překážka, která paprsek světla zablokuje.



Obrázek 2.5: Schéma ablace povrchu sněhu. Rozptyl světla z bodu x_1 do intervalu mezi x a $x + dx$ závisí na úhlu $d\theta$. Vektor $\mathbf{p}$ směřuje z bodu x_1 do bodu x a vektor dl je normála na $\mathbf{p}$ tak, že $d\theta = dl/p$. Vektor $\mathbf{n}$ je normála k povrchu v bodě x a vektor $d\mathbf{s}$ je inkrement podél povrchu mezi x a $x + dx$. Schéma bylo převzato z [7].

2.4.3 Celkový model

Celkový model pak kombinuje rovnice světelné odrazivosti 2.11 a ablace 2.1 takto:

$$\frac{\delta h}{\delta t} = -\frac{\alpha I}{L}\mathcal{I}(x) + D\frac{\delta^2 h}{\delta x^2}, \tag{2.12}$$

kde je definován integrál

$$\mathcal{I}(x) = \frac{1}{\pi}\int_{line\,of\,sight} \frac{dx_1(\Delta h - h'(x)\Delta x)}{\Delta h^2 + \Delta x^2}. \tag{2.13}$$

Intenzita slunce určuje charakteristickou rychlost ablace IL^{-1}, kde L je latentní teplo na objemovou jednotku. Spojením této rychlosti a difúzního koeficientu D dává rozměr nejmenšího možného objektu

$$\bar{\ell} = \frac{DL}{I} \tag{2.14}$$

a čas, za který vznikne

$$\bar{t} = \frac{DL^2}{I^2} \tag{2.15}$$

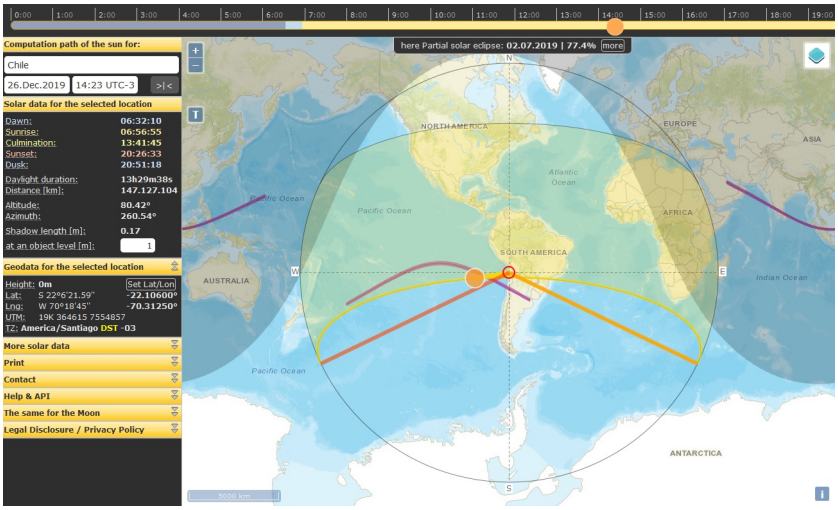
Po dosazení jsou získány hodnoty pro parametr rozměru $\bar{\ell} = 0,25\,mm$ a času $\bar{t} = 25\,s$.

Pro sublimaci sněhu je latentní teplo sedminásobně vyšší, což ve výsledku znamená menší poměr tání $I/L = 1,4 \times 10^{-3} cm\,s^{-1}$, větší parametr rozměru $\bar{\ell} = 1,75\,mm$ a času $\bar{t} = 1225\,s$.

2.5 Pozice Slunce na obloze

Kajícníci směřují svými vrcholky k poloze Slunce na obloze v pravé poledne, kdy je záření nejsilnější, podle [7] dochází k ablaci nejvíce po čtyři hodiny, kdy je sluneční záření nejintenzivnější. Rovněž je třeba znát úhel dopadu slunečních paprsků pro výpočet jejich prvotního odrazu od povrchu. To znamená, že je klíčové určit pozici Slunce v období ablace, navíc pozice Slunce se v průběhu roku mění.

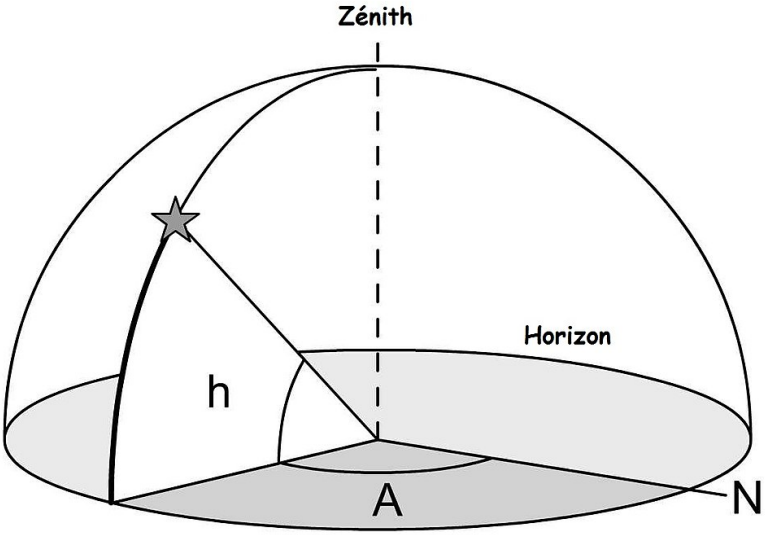
Nástroj SunCalc je dostupný na webové stránce[2] nebo jako aplikace pro OS Android. Dokáže pro zadané místo a čas určit astronomický azimut a výšku, a také pro dané místo určí údaje o východu a západu Slunce i čas, kdy zde aktivita Slunce kulminuje.



Obrázek 2.6: Ukázka výstupu internetové stránky SunCalc.org. V levém sloupci jsou potřebná data, zejména *Altitude* značící výšku a *Azimuth*, astronomický azimut.[12]

Další možnost je využít HelioWatcher[3], systém primárně určený k automatickému nastavování polohy solárních panelů pomocí GPS, jehož funkce napsané v jazyce C mohou také poskytnout potřebné údaje.

[2] https://www.suncalc.org
[3] https://github.com/sciguy14/HelioWatcher



Obrázek 2.7: Obzorníkové souřadnice. Astronomický azimut A vyjadřuje úhel mezi severním směrem a směrem k bodu na vodorovné ploše, kde se promítá pozorované nebeské těleso. Výška h určuje úhel mezi vodorovnou rovinou a polopřímkou, která prochází z místa pozorovatele pozorovaným tělesem.[16]

Kapitola 3

Návrh řešení

Na základě teoretických znalostí z předchozí kapitoly je zde popsán návrh programu, který bude simulovat vznik Kajícníků. Nejprve zde budou uvedeny použité technologie a následně se kapitola bude věnovat samotnému návrhu.

3.1 Použité technologie

Jako první krok je nutné vybrat vhodné nástroje pro řešení daného problému, a zvolit programovací jazyk. Tento výběr může velmi ovlivnit složitost dosažení výsledku i jeho výsledky.

3.1.1 Implementace modelu

Pro implementaci bude použitý programovací jazyk C++. Tento jazyk byl vybrán zejména kvůli jeho univerzálnosti a rozšíření, jeho velká uživatelská základna poskytuje množství znovupoužitelného kódu v podobě volně dostupných knihoven a podporu při řešení problémů. Implementace bude provedena ve formě projektu vývojového prostředí Code::Blocks, jehož výhodou je podpora více platforem i překladačů, také se jedná o svobodný, open-source software.

Původně bylo v návrhu implementace počítáno s využitím knihovny pracující s objemovou reprezentací objektů, tedy že vstupem i výstupem bude volumetrický 3D model. Výhodou tohoto řešení by byla zejména možnost dalšího rozvoje práce směrem k renderování. Nicméně knihovna OpenVDB volně dostupná z https://www.openvdb.org/, která byla původně pro tento účel zvolena, je poměrně složitá, rozměrná a vyžaduje velké množství závislých knihoven pro svoji funkčnost, což je pro zvolený model vcelku zbytečné. Tento model je totiž ve své podstatě jednorozměrný, vstupní data musejí být navzorkována do jednotlivých řezů, a až výsledky jsou opět spojeny do trojrozměrné reprezentace.

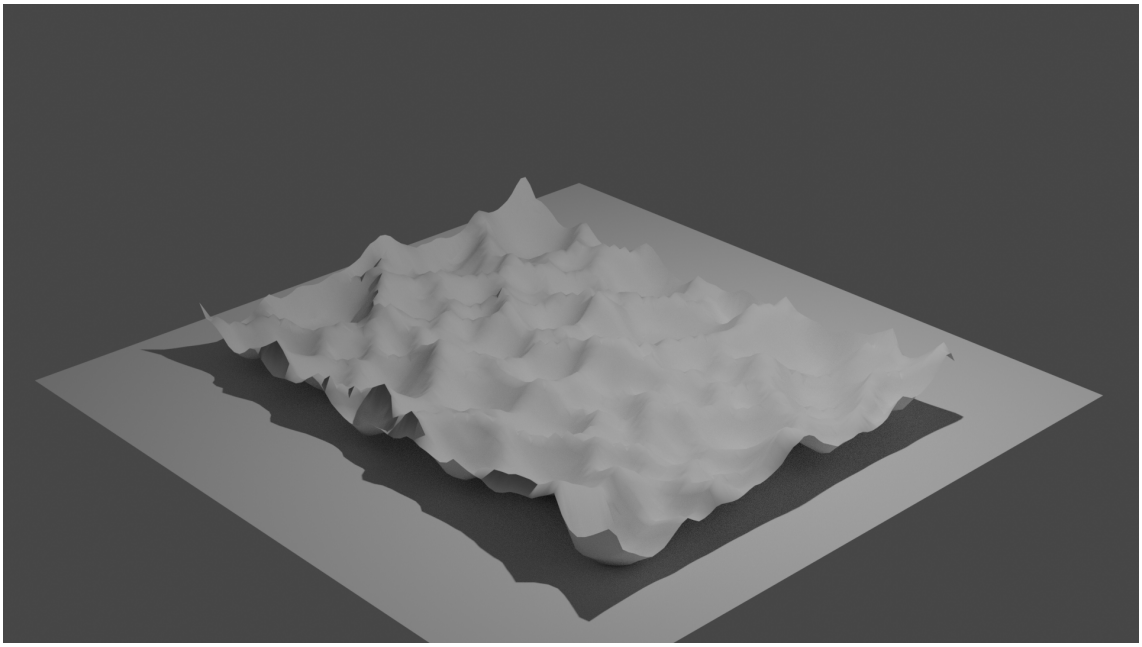
Jelikož výpočty jednotlivých řezů jsou na sobě nezávislé, nabízí se zde značný prostor k využití plné výpočetní síly stroje a úspoře času pomocí paralelizace. V tomto případě bylo zvoleno volně dostupné a rozšířené řešení v podobě rozhraní OpenMP. To je multiplatformní soustava direktiv pro paralelní programování v programovacích jazycích Fortran, C a C++. OpenMP se získá jako součást překladače daného jazyka a využívá direktivy *#pragma omp*, které specifikují jeho fungování, zejména zda bude daný blok kódu možné provádět paralelně, či nikoli. V případě, že použitý překladač OpenMP nepodporuje, tyto direktivy jsou typicky ignorovány a kód je stále přeložitelný a plně funkční, i když ne paralelizovaný.

3.1.2 Vstup a výstup

Jako vstup a výstup samotného programu byly zvoleny výškové mapy v podobě rastrových obrázků ve stupních šedi, kde barva v rozmezí 0-255 značí výšku terénu. Určitou nevýhodou této reprezentace je nemožnost zachytit dutiny nebo horizontální výčnělky či převisy, ale vzhledem k povaze materiálu, ze kterého se Kajícníci tvoří i jejich samotné tvorbě by tato schopnost neměla být potřeba. Čtení ani zápis do souboru rastrového obrázku není nativní schopností jazyka C++, nicméně existuje mnoho dostupných knihoven, které potřebné implementace poskytují, a to od rozsáhlých projektů typu OpenCV až po jednoduché knihovny poskytující pouze nutné základní funkce, zato ale přenositelné. Jednou z takových je knihovna *tinyexr*, která byla zvolena.

3.1.3 Vizualizace

Jelikož výstupem programu bude opět obrázek výškové mapy, je třeba ještě vybrat vhodný nástroj pro vizualizaci této výškové mapy v trojrozměrném prostoru, což je vhodné pro porovnání výsledku s reálnými daty. Pro tento účel byl zvolen nástroj Blender. Blender je open-source software pro vykreslování a modelování 3D grafiky. Jeho hlavní výhodou je kromě distribuce zdarma i dostupnost na různé platformy a široká uživatelská základna poskytující návody na používání a řešení možných problémů, a také množství rozšíření a pluginů. Rovněž podporuje použití rastrového formátu OpenEXR, který byl zvolen pro výškové mapy.



Obrázek 3.1: Ukázka výstupu programu Blender.

3.1.4 OpenEXR a tinyexr

Knihovna *tinyexr* je jednoduchá knihovna v jazyce C++ ve formě jediného hlavičkového souboru, která implementuje funkce a struktury pro práci se soubory typu OpenEXR, jejímž

autorem je Syoyo Fujita. Knihovna je dostupná na https://github.com/syoyo/tinyexr. Její výhodou je oproti standardní implementaci knihovny OpenEXR pro práci s tímto typem souborů zejména jednoduchá přenositelnost programu, který ji využívá. Není třeba totiž žádná instalace a kompilace, hlavičkový soubor knihovny se jednoduše přidá ke zdrojovým souborům programu a je spolu s ním i překládán.

OpenEXR je rastrový formát souboru s technologií HDR (High Dynamic Range). To je technologie, která umožňuje rozšířenou reprodukci detailů v tmavých i světlých částech snímané scény zároveň. Standard tohoto formátu společně se softwarovými nástroji vytvořila společnost ILM (Industrial Light & Magic) v roce 1999 a zveřejnila ho pod licencí svobodného softwaru v roce 2003. Dokumentace a nástoje jsou dostupné na adrese https://www.openexr.com/.

Na rozdíl od běžně používaných formátů OpenEXR nepoužívá pro uchovávání hodnoty odstínu barevného kanálu celá čísla, ale typy s plovoucí řádovou čárkou. Běžná je implementace s šestnáctibitovou reprezentací, ale jsou i jiné. Hlavní výhoda spočívá v daleko větším množství barev, které lze uchovávat. Rovněž lze ukládat hodnoty větší a menší, než jsou zobrazitelné hodnoty. Vhodnou úpravou těchto dat pro zobrazení lze následně dosáhnout lepšího kontrastu scény, výrazně prokreslenějších barevných odstínů a větší hloubky obrazu. Scény tak působí živějším dojmem a více odpovídají realitě, protože lidské oko si dovede s výrazně kontrastními scenériemi poradit lépe než optické přístroje. Viditelné hodnoty jsou ve formátu OpenEXR hodnoty 0 (černá) až 1 (bílá).

Pro editaci výškových map budou použity programy GiMP 2.10 a Adobe Photoshop CS3 s volně dostupným plug-inem pro editaci souborů typu .exr Exr-IO[1] a výsledky budou vizualizovány v Blenderu. GiMP a Blender jsou programy volně šiřitelné pod licencí GNU GPL, Photoshop je komerční software. Jelikož zejména Blender dokáže pracovat pouze s rozsahem hodnot 0-1, bude třeba uchovávané hodnoty škálovat uvnitř programu před prováděním výpočtu.

3.2 Návrh programu na simulaci vzniku Kajícníků

3.2.1 Vstup

Prvním nutným úkonem programu je načíst vstupní obrázek. Cesta k jeho umístění na disku bude předána jako parametr při spuštění programu. Zde je předpokládán existující soubor ve formátu .exr ve stupních šedi. Je třeba nejprve ověřit, že tyto požadavky jsou splněny, jelikož jsou nutné pro správnou práci se zdrojem. Další předpoklady jsou, že obrázek skutečně obsahuje výškovou mapu a že je jako mapa orientován, tedy sever nahoře a východ vpravo. Tyto vlastnosti ale program neověřuje a pouze předpokládá jejich pravdivost.

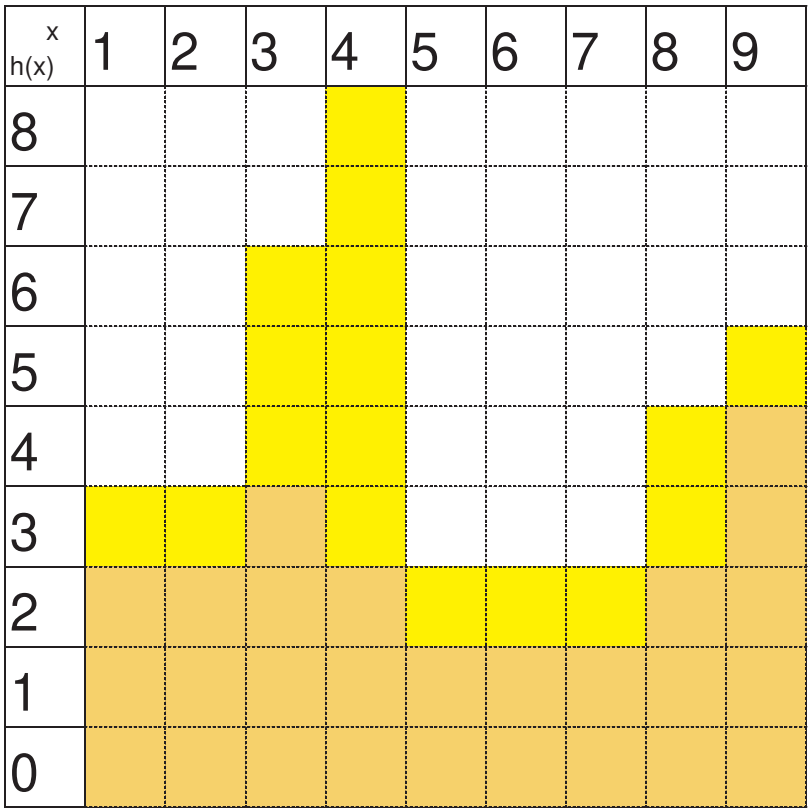
Data obsažená ve vstupním souboru je nyní třeba převést do podoby, která je vhodná pro zpracování ve zvoleném modelu. Jelikož se jedná o model jednodimenzionální, je nutné vytvořit sérii rovnoběžných řezů terénem, který bude navzorkován. Rastrový soubor sám o sobě už obsahuje vzorkovaná data jak v podobě vhodné pro řezy, tak i jako jednotlivé vzorky. Řezem je jeden řádek pixelů, souřadnice y je konstantní, jedním vzorkem je potom pixel. Ten jednak vyjadřuje souřadnici x v podobě svého pořadí v řádku, kde 0 je nejzápadnější vzorek, a zároveň také jeho barva označuje výšku sněhu v daném bodě, v případě trojrozměrného zobrazení se jedná o souřadnici z. Jednotlivé vzorky-body si tak lze pro potřeby programu představit jako kvádry odpovídající voxelům v trojrozměrném objektu,

[1] https://www.exr-io.com/

jehož délka a šířka je rovna vzdálenosti mezi 2 sousedními vzorky. Výška kvádru je 1. Zde je vidět vztah s původně zamýšlenou objemovou reprezentací.

3.2.2 Popis povrchu

Takto jsou nyní známy body povrchu, jejichž svrchní stěna je vystavena Slunečnímu záření. Toto ale nejsou všechny body, na které záření může dopadat. Další body se mohou nacházet situované pod těmito body, přičemž budou mít odhalenou jednu nebo více bočních stěn. V podstatě je třeba vstupní pole výšek převést na pole, kde jsou uvedeny všechny body v takovém pořadí, kudy by vedla diskretizovaná funkce $y = h(x)$ daného řezu. Pro další zpracování je zapotřebí tyto body identifikovat. Bod je jednoznačně definován svou souřadnicí x a $h(x) = y$. Jelikož je ale pro potřeby výpočtu znát zejména povrch této struktury,



Obrázek 3.2: Vizualizace příkladu povrchu. Žlutě jsou značeny vzorky na které může dopadat světlo. Vstupem pro tuto reprezentaci by bylo pole {3, 3, 6, 8, 2, 2, 2, 4, 5}

.

může dojít k situaci, kdy má bod dvě boční stěny vystaveny záření, ale nacházejí se na nesouvislých částech povrchu. Na obrázku 3.2 je takovým bodem bod $[4; 7]$, jehož dvě boční stěny jsou z pohledu povrchu odděleny povrchem bodu $[4; 8]$. Z předchozích informací vyplývá, že při popisu povrchu pomocí jednotlivých bodů může dojít k situaci, že pro jednu souřadnici x bude více bodů s různými souřadnicemi $h(x)$ a za určitých okolností se jednotlivé body dokonce budou opakovat. Proto je třeba vstupní pole pole převést na pole, kde jsou uvedeny všechny body v takovém pořadí, kudy by vedla funkce $y = h(x)$. V případě

příkladu na obrázku 3.2 bude výstupem pole dvojic {[1;3], [2;3], [3;4], [3;5], [3;6], [4;7], [4;8], [4;7], [4;6], [4;5], [4;3], [5;2], [6;2], [7;2], [8;3], [8;4], [9;5]}.

S povrchem souvisí ještě další potřebný údaj pro výpočty simulace, a to je normála k povrchu v daném bodě. Normála je přímka kolmá na tečnu, která prochází daným bodem. O tečně je známo, že její směrnice je rovna hodnotě derivace funkce v bodě dotyku, tedy: $f'(x_0) = \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0}$. Pro program bude nutné znát normálu jako vektor kolmý na povrch, který směřuje od povrchu do prostoru, kde je vzduch. Protože derivace je definována jako rozdíl, komponenty vektoru směrnice se určí jako rozdíly hodnot $x$ových a $y$ových souřadnic předchozího a následujícího bodu, normála pak bude kolmá na tento vektor. Zde se opět projevuje výhoda a nutnost ukládání bodů vysvětlena v předchozím odstavci, jelikož bod $[4; 7]$ z obrázku 3.2 bude mít pro každou ze svých dvou instancí v popisu povrchu jinou normálu. Pro vysvětlení, bod $[4; 8]$ ve stejném obrázku bude mít normálu $(0; 1)$, jelikož se jedná o lokální maximum, a tudíž derivace v tomto bodě musí být rovna 0. Oproti vzorci je třeba počítat i se situací, že tečna v bodě bude svislá, jako v případě obrázku 3.2 například u bodu [4;5] a dalších.

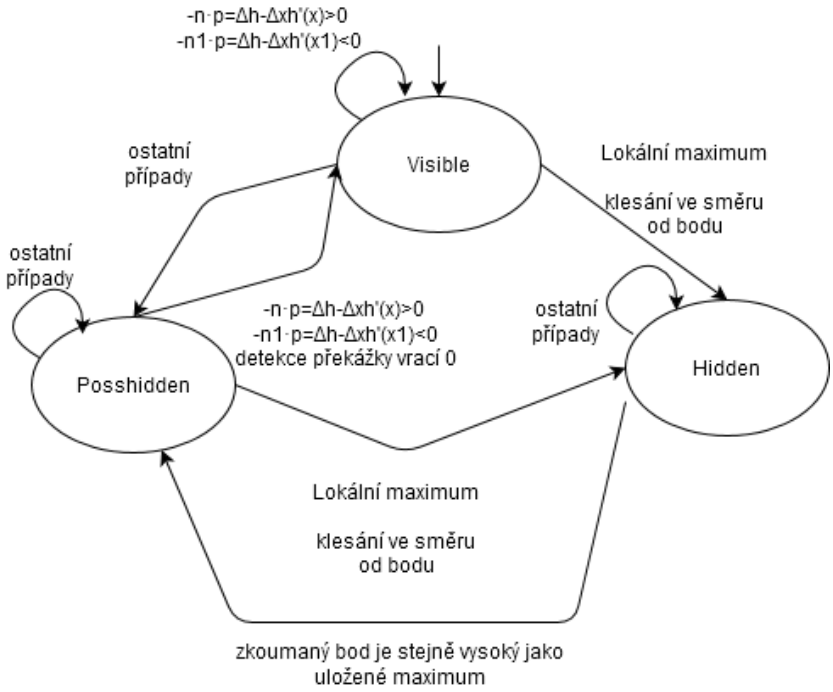
3.2.3 Viditelnost bodů

Je třeba určit body, ze kterých se bude odrážet světlo do zvoleného bodu. Jedna z nutných podmínek už ubyla uvedena v předchozí kapitole, a sice, že body x_1 a x jsou ve vzájemné linii pohledu, pokud platí, že výsledek skalárního součinu vektoru $\mathbf{n}$ kolmého k povrchu v bodě x a vektoru $\mathbf{p}$ směřujícího z bodu x_1 do x (viz. schéma 2.5) je menší, než 0. Pro potřeby výpočtu ze známých proměnných lze tento vztah vyjádřit jako $-n \cdot p = \Delta h - \Delta x h'(x) > 0$. Tento vztah v podstatě vyjadřuje, že bod x musí být přivrácený k paprsku, který na něj dopadá. Obdobný vztah musí mít s vektorem $\mathbf{p}$ i bod x_1, ze kterého paprsky vycházejí. Ale jelikož v tomto případě má vektor $\mathbf{p}$ oproti předchozímu případu opačný směr, je třeba toto v podmínce zahrnout. Tedy pokud n_1 je vektor kolmý na směrnici v bodě x_1, poté platí, že $-n_1 \cdot p = \Delta h - \Delta x h'(x_1) < 0$.

Tyto dvě podmínky vyjadřují, že oba body jsou na vzájemně přivrácených plochách, nicméně nevyjadřují omezení, že se mezi nimi nenachází překážka, která by výhled z jednoho na druhý blokovala. Překážka bude v popisu povrchu detekovatelná lokálním maximem. Lokální maximum má derivaci rovnou 0, a jelikož tato vlastnost bude využita v této části, je nutné během popisu povrchu (předchozí část) tuto vlastnost zajistit. Derivace rovna nule ale platí i pro lokální minimum, proto je nutné detekovat ještě další vlastnost povrchu a sice klesání. Z popisu povrchu vyplývá, že směrnice bodu bude mít x-ovou souřadnici kladnou nebo rovnou nule. To znamená, že je zapotřebí uvážit směr, kterým je povrch procházen. Při průchodu směrem vpravo x roste, a tím pádem budou směrnice mít odpovídající směr. V tomto případě je hledáno místo, kde je derivace rovna nule a existuje následující bod, jehož směrnice má zápornou y-ovou souřadnici. Lokální extrém následovaný klesáním je lokální maximum. V případě, že je ale povrch procházen směrem doleva, směrnice vedou proti směru průchodu a naopak je hledaný bod ten s nulovou derivací, jehož předcházející bod (následující ve směru průchodu), jehož y souřadnice je kladná.

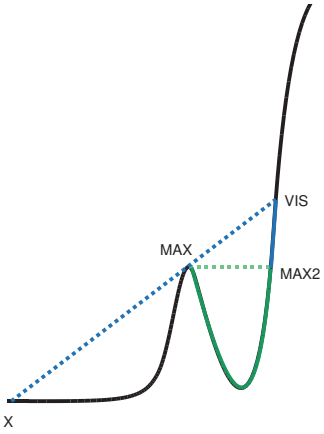
Ne všechny lokální maxima po cestě ale nemusí nutně blokovat výhled mezi body. Minimální podmínka pro překážku je, že je alespoň tak vysoká, jako jeden z bodů. Nižší maximum totiž viditelnost mezi body ovlivní pouze částečně. Rovněž maximum nemusí zastiňovat úplně všechny následující body v řezu. Byl navržen algoritmus, který projde všechny body řezu v obou směrech od bodu, od kterého jsou viditelné body určovány.

Tento algoritmus je inspirován konečným automatem, na rozdíl od něj ale nemá koncový uzel, ale končí projitím všech bodů. Jeho diagram je zobrazen na obrázku 3.3

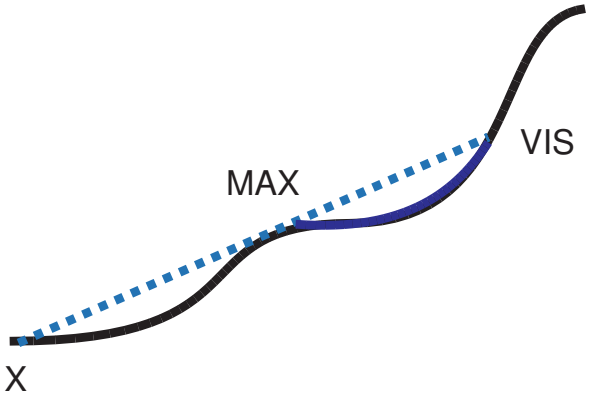


Obrázek 3.3: Diagram stavů v programu a situací způsobujících přechody mezi nimi

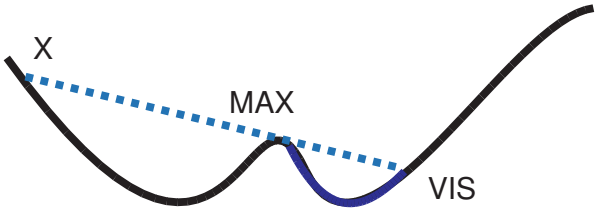
Program začíná ve stavu *Visible* a prověřuje bod sousedící s bodem X, pro který jsou určovány viditelné body. Jelikož rozdíly mezi vyšetřováním bodu nalevo a napravo od tohoto bodu již byly probrány dříve v této kapitole, tato část se bude věnovat obecnému bodu. V tomto stavu usuzuje, že řešený bod je viditelný, prověřuje zda skutečně platí, že plochy jsou k sobě přivrácené. Pokud tomu tak je, ukládá bod jako viditelný. Pokud narazí na bod vyšší než bod X, kde povrch začíná klesat, uloží tento bod jako maximum a přepne se do stavu *Hidden*. V tomto stavu usuzuje, že vyšetřovaný bod bude od bodu X zastíněný, a proto se omezuje na hledání bodu, který bude mít alespoň takovou výšku, jako dříve nalezené maximum. V případě nalezení takového bodu se přepíná do stavu *Posshidden*. Do tohoto stavu se rovněž přepíná ze stavu *Visible* v případě, že nejsou splněny podmínky ani pro setrvání ve stavu *Visible*, ani pro přepnutí do stavu *Hidden*. Stav *Posshidden* je nejobecnější stav, kdy je třeba hlouběji prověřit stav řešeného bodu. Toto prověření se podobá stavu *Visible*, takže rovněž prověřuje přivrácenost ploch a zda není detekována překážka. Oproti tomu v případě přivrácenosti stran ještě vysílá paprsek mezi X a řešeným bodem a v každém místě tohoto paprsku porovnává jeho výšku s výškou terénu v daném místě. Pokud je paprsek pod terénem, v cestě je překážka a body na sebe nevidí. Pokud paprsek úspěšně dorazí až do cíle s výškou terénu všude nižší, body jsou vzájemně viditelné. Program tudíž ukládá bod jako viditelný a přepíná se do stavu *Visible*. To je umožněno tím, že vzhledem k způsobu vstupu je vyloučena přítomnost dutiny nebo oblouku, který by dokázal omezit viditelnost shora. Pro další vysvětlení průběhu tohoto algoritmu jsou připraveny názorné příklady 3.4, 3.5 a 3.6 znázorňující situace ve tvaru terénu vedoucí k přepínání mezi stavy.



Obrázek 3.4: Program prochází povrch od bodu X ve stavu *Visible* (černou barvou). V bodě *MAX* jsou splněny podmínky pro přechod do stavu *Hidden* (zelenou barvou). Program prochází následující body a hledá bod stejně vysoký jako *MAX*, ten najde v bodě *MAX2*. Zde se přepne do stavu *Posshidden* (modrou barvou) a jelikož jsou plochy k sobě přivrácené, posílá mezi vyšetřovanými body a X paprsky. V bodě *VIS* poprvé nedetekuje překážku a přepne se tedy opět do *Visible*. Je možné, že už před přechodem do *Hidden* jsou nalezeny body, které k sobě nejsou přivrácené, a automat je proto přepnut do stavu *Posshidden*, což ale vede ke stejnému výsledku. Vzniku tohoto stavu se věnuje obrázek 3.5 a přepnutí z *Posshidden* do *Hidden* odpovídá tomu popsanému zde.



Obrázek 3.5: Program prochází povrch od bodu X ve stavu *Visible* (černou barvou). V bodě *MAX* přestanou být plochy k sobě přivrácené, což vede k přechodu do *Posshidden*, nicméně povrch je neklesající. Po čase začnou být k sobě body opět přivrácené, což vede k prověřování vysíláním paprsků. Úspěch v bodě *VIS* vede k přepnutí do *Visible*.



Obrázek 3.6: Program prochází povrch od bodu X ve stavu *Visible* (černou barvou). V bodě *MAX* přestanou být plochy k sobě přivrácené, což vede k přechodu do *Posshidden*. Povrch sice klesá, ale bod není vyšší, než X, proto nedojde k přechodu do *Hidden*. Body jsou prověřovány na přivrácenost a následně i vysíláním paprsků, úspěch v bodě *VIS* opět vede k přepnutí do *Visible*.

3.2.4 Výpočet energie

Z matematického modelu je znám vzorec pro výpočet změny výšky za čas 2.12. Jelikož tento program dokáže pracovat pouze s diskrétními daty, je třeba část vzorce obsahující integrál adekvátně upravit a sice:

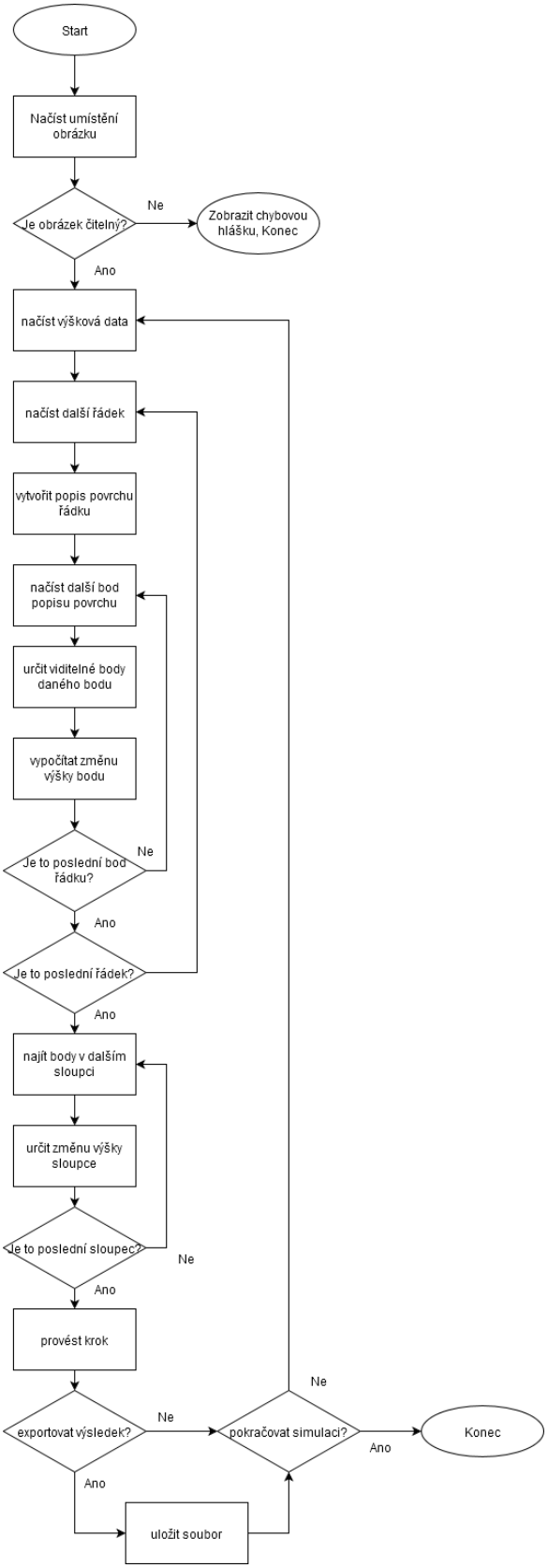
$$\mathcal{I}(x) = \frac{1}{\pi} \sum_{line\,of\,sight} \frac{\Delta h - h'(x)\Delta x}{\Delta h^2 + \Delta x^2}, \tag{3.1}$$

kde *line of sight* vyjadřuje množinu bodů popsaných uspořádanými dvojicemi $[x_i, h(x_i)]$, které jsou viditelné z bodu x. Získání této množiny bylo předmětem předchozí podkapitoly. Nyní je známa rychlost změny výšky pro každý z bodů, na který dopadá Sluneční záření, nebo jeho odraz. Problém je, že původní model počítá se vstupem, který lze definovat matematickými funkcemi, zatímco tento model má oproti tomuto dvě hlavní odlišnosti:

1. je dovoleno pro jedno x mít více různých hodnot y, aby šlo popsat strmější klesání, než je 45°.

2. jeden bod je na více místech povrchu a tak má dvě rychlosti klesání.

Obě tyto odlišnosti je třeba řešit a určit jednu rychlost pro celou výšku sloupce. Jako nevhodnější se jeví průměr nebo medián rychlostí jednotlivých bodů ve sloupci. Každý sloupec má nyní svou současnou rychlost klesání výšky $h(x)$ a poslední částí zde je provedení časového posunutí, kdy každý ze sloupců po daný čas svou rychlostí klesá. Po dokončení kroku je terén změněn. V rámci této změny bude třeba zkontrolovat a případně normalizovat výsledky. Výstupní soubory typu .exr ukládají hodnoty s plovoucí řádovou čárkou, takže odpadá nutnost zaokrouhlování na celá čísla, které by bylo nutné u obvyklých rastrových souborů. Další věc je normalizace do rozsahu použitého výstupního souboru. Vzhledem k předchozímu návrhu se očekává, že horní okraj není třeba upravovat, jelikož vstupem je také obrázek, což znamená, že vstupní hodnoty jsou v daném rozsahu. Samotný model pak provádí pouze snižování terénu. Může se ale stát, že výsledky simulace povedou do záporných čísel. Ty nelze v obrázkovém souboru jakkoli zobrazovat, a tak budou tyto výsledky převedeny na na nulovou hodnotu. Je možné vygenerované výšky sloupců vložit do formátu obrázku a vyexportovat pro zobrazení výsledku v programu Blender nebo pokračovat s dalším krokem simulace. V tomto případě se program vrací k popisu povrchu a proces se opakuje.



Obrázek 3.7: diagram návrhu průběhu programu.

.

Kapitola 4

Implementace

Samotná implementace simulace se skládá ze čtyř souborů: „main.cpp“, „functions.h“, „tinyexr.h“ a „missingfunction.h“. Soubor „tinyexr.h“ obsahuje kódy knihovny *tinyexr*, která zajišťuje práci s bitmapovými soubory typu OpenEXR. „missingfunction.h“ obsahuje implementaci funkce strdup, kterou knihovna *tinyexr* využívá, ale není podporována použitým překladačem jazyka C++. Aby nedošlo ke konfliktu s existující funkcí v jiných překladačích, je funkce nazvána my_strdup.

4.1 Hlavní funkce

„main.cpp“ obsahuje hlavní funkci. Hlavním úkolem této funkce je zjištění a kontrola argumentů programu, načtení vstupního souboru, řízení simulace a generování výstupních dat. Funkce nejprve zkontroluje počet parametrů zadaných při spuštění. Jsou čtyři parametry, které jsou požadovány:

1. Název vstupního souboru a případná cesta k němu. Zde je očekáván řetězec znaků.
2. Délka jednoho kroku v sekundách. Zde musí být zadáno přirozené číslo. Vhodná hodnota zde je například 3600, tedy 1 hodina. Při vyšších hodnotách dochází k chybám na velmi strmých plochách.
3. Celkový počet kroků. Opět je požadováno přirozené číslo.
4. Počet kroků mezi generováním výstupů. Pokud není třeba generovat výstup u každého kroku simulace, ale stačí každý x-tý, potom hodnota x bude tímto parametrem, například pokud je cílem generovat výstup jednou za den simulace s krokem 1 hodina, tento parametr je třeba nastavit na 24. Je vhodné, aby tento parametr byl dělitelem celkového počtu kroků. Stejně jako u předchozích parametrů je požadováno celé číslo.

Funkce zkontroluje, zda jsou parametry zadány a zda číselné parametry splňují podmínky. Následně je řetězec předán do funkce pro načtení souboru `LoadEXR(float **out_rgba, int *width, int *height, const char *filename, const char **err);`. Tato funkce má v sobě už implementovanou kontrolu, zda soubor existuje. V případě úspěšného nalezení a dekódování ukládá bitmapový obrázek v podobě jednorozměrného pole hodnot typu float o délce šířka*výška*4 (kanály červená, modrá, zelená a alfa-průhlednost) do out_rgba a vrací hodnotu 0. Rovněž je připraven řetězec pro ukládání výstupních souborů. Pokud byl řetězec parametru delší, než čtyři znaky, lze předpokládat, že obsahuje koncovku .exr.

V tomto případě jsou poslední čtyři znaky (což je předpokládaná koncovka) odstraněny pro následné generování jmen nových výstupních souborů. Výšková mapa je nyní v jednorozměrném poli hodnot typu float s redundantními hodnotami, všechny barevné kanály mají totiž v případě šedotónového obrazu stejnou hodnotu a alfa má hodnotu 0. Pro názornější implementaci bylo zvoleno tyto data uchovávat jako dvourozměrné pole hodnot typu double. Pole s daty je v rámci vnořených cyklů celé projito a významné hodnoty z něj překopírovány do dvourozměrného pole. Zde je zároveň provedeno škálování hodnot, jelikož do souboru typu .exr nelze v rámci programu GiMP 2.10 vkládat hodnoty mimo viditlené spektrum, tedy mimo interval <0,1>. Je rovněž třeba zde poukázat na to, že i program Blender, který je následně použit na renderování výsledku, tyto hodnoty stejně ořezává na daný interval. Omezení na hodnoty v intervalu <0,1> tak vyžaduje úpravu rozsahu uvnitř programu. Jako řešení bylo zvoleno škálování. Problém by šel řešit i tak, že by rozměr tloušťky vrstvy byl definován například jako 0-1 km, škálování ale v závislosti na zvolené konstantě nabízí variabilnější možnosti v přesnosti vstupního souboru z programu GiMP 2.10 (vysvětleno v kapitole 5.1). Škálování je prováděno násobením s konstantou *resizer*, která je rovněž definována na začátku tohoto souboru. Následuje uvolnění jednorozměrného pole z paměti, jelikož to už nemá pro simulaci smysl uchovávat, dále jsou nastaveny parametry pro běh simulace.

Simulace samotná se skládá z dvou volání funkce `RunVector(vector<double> &v, int time)` uvnitř vnořených cyklů. Vnější cyklus počítá celé kroky, vnitřní cykly počítají jednotlivé řádky nebo sloupce. Funkce `RunVector` zastřešuje krok simulace v jednom řezu povrchu. Na vstupu je jí vložen vektor hodnot typu double obsahující výšky jednotlivých bodů a délka kroku. Výstup, neboli nové výšky bodů o jeden krok simulace později jsou opět uloženy do vstupního vektoru. Tato funkce je zde využita dvakrát. Nejprve bere jako parametr vektor, který je uložený ve vektoru reprezentujícím dvourozměrnou matici. Zde se jedná o přístup k jednotlivým řádkům této matice. Přístup ke sloupcům matice je trochu složitější, a je implementován pomocí funkce `getVertical(vector<vector<double>> matrix, int width, int height, int col, vector<double> &reslt)`, která překopíruje data sloupce do pomocného vektoru, ten je následně použit pro výpočet a výsledek poté zpět uložen do matice funkcí `saveVertical(vector<vector<double>> &matrix, int width, int height, int col, vector<double> vect)`. Lze říct, že funkce `getVertical` a `saveVertical` fungují zrcadlově. Uvnitř vnějšího cyklu ještě běží počítadlo kroků, které říká, zda je čas vytvořit výstupní soubor. V případě, že dojde na hodnotu stanovenou posledním parametrem, tak se vynuluje, nastaví název vytvořeného souboru, vezme data v současné době uložená ve dvourozměrném vektoru a vytvoří z nich jednorozměrné pole hodnot. To je deškálované, aby hodnoty odpovídaly vstupnímu souboru, a následně konvertované na typ float. Tyto dvě sekce programu jsou připravené pro paralelizaci pomocí knihovny OpenMP, současně tak může probíhat výpočet několika řezů na různých jádrech procesoru, což významně urychluje běh programu. V případě práce se sloupci je zde pro jistotu nastavena kritická sekce, aby nedocházelo k chybě přepisování hodnot, protože data jsou uchovávána ve struktuře vektorů reprezentující řádky mapy.

Kód programu pokračuje voláním funkce `SaveEXR(const float *data, const int width, const int height, const int components, const int save_as_fp16, const char *filename, const char **err)`, která dané pole hodnot float správně uloží do souboru typu .exr. Nastavení jména souboru spočívá v připojení řetězce _X.exr, kde X je pořadové číslo výstupního souboru, za řetězec vzniklý z prvního parametru. například pro vstupní soubor Text.exr budou vznikat výstupní soubory Test_1.exr, Test_2.exr, atd.

4.2 Výpočet kroku simulace v řezu

Funkce `RunVector(vector<double> &v, int time)` slouží k zapouzdření implementace výpočtu jednoho kroku simulace na jednom řezu. Uvnitř této funkce je popsán povrch řezu, pro každý bod povrchu určeny viditelné body, proveden výpočet rychlosti klesání pro každý bod a nakonec vypočten časový posun, což je změna terénu za jeden krok simulace, a výsledek zpět uložen do matice reprezentující povrch.

Podle části 3.2.2 se popis povrchu skládá ze dvou úkonů. Napřed je třeba identifikovat všechny body s alespoň jednou viditelnou stěnou, a u těchto bodů popsat tvar povrchu v nich pomocí směrnice. Identifikaci bodů řeší funkce `active_points (vector<double> &vect, vector<points> &Pts)`. Ta prochází vektor *vect* a nejprve porovná současný vyšetřovaný a předchozí bod. Pokud má současný bod menší výšku než bod předchozí, vytváří funkce body, které jsou pod předchozím bodem, tedy mají jeho souřadnici ve vodorovném směru, a jejich svislé souřadnice jsou mezi výškami těchto dvou bodů a jsou z oboru celých čísel, která po sobě následují sestupně. Pokud je rozdíl mezi výškami bodu menší nebo roven 0, body se dotýkají a není třeba tvořit nový bod. Pro případ, že současný bod má vyšší výšku, než ten předchozí, postup je podobný, pouze vodorovná souřadnice odpovídá současnému bodu a svislé souřadnice rostou. Výjimkou je první, ten nejlevější, bod vektoru. Ten nemá žádný předchozí bod. Případné body ležící pod ním budou identifikovány při inspekci druhého bodu ve vektoru. Nakonec do vektoru vloží i současný bod a přesunuje se na bod následující. Struktura typu *points* obsahuje dvě hodnoty typu *double*, a to hodnotu na vodorovné souřadnici *x_axis* a hodnotu na svislé souřadnici *y_axis*. Výsledkem této funkce je tak vektor *Pts*, který obsahuje struktury typu *points* reprezentující body řezu tak, jak jdou za sebou zleva doprava.

Funkce `GenerateSlopeVector(vector<points> Pts)` vrací vektor hodnot typu *points*, kde každá hodnota odpovídá vektoru směrnice povrchu v bodě uloženém ve vektoru *Pts* na stejné pozici. Zde byla použita stejná struktura, jako u vektoru uchovávajícího body, ale významově se jedná o dvourozměrné vektory s vodorovnou a svislou komponentou. Směrnice povrchu v daném bodě je tím pádem vektor, který má počátek v daném bodě (uloženém ve vektoru *Pts*) a směr definovaný těmito komponentami. Pro výpočet těchto komponent je použita funkce `Slope(vector<points> Pts, int PointNum, points &pointer)`. Ta pro bod z vektoru *Pts*, kde je na *PointNum*-tém místě (v C++ samozřejmě počítáno od 0) vypočítá komponenty směrnice a uloží je do struktury *pointer*. Vodorovná komponenta je určena jako rozdíl vodorovných souřadnic následujícího a předchozího bodu, svislá je určena obdobně. Výjimkou je první a poslední bod vektoru. Směrnice u prvního je určena jako rozdíl druhého a prvního, poslední jako rozdíl posledního a předposledního, jelikož tyto body předchozí, respektive následující bod nemají.

Program pokračuje voláním funkce `getHeight(vector<points> Pts,vector<points> Slopes)`, která ovšem téměř ihned po nastavení lokálních proměnných volá `getAblation (vector<points> Pts, int PointNum, vector<points> Slopes)`, která rovněž začíná voláním `NewgetVisible (vector<points> Pts, int PointNum,vector<points> Slopes, vector<int> &Reslt)`. Pro objasnění toku programu je proto vhodné začít vysvětlení výpočtu rychlosti klesání v bodu touto funkcí, která má za úkol určit body, ze kterých se odráží světlo do tohoto zvoleného bodu (dále viditelné body). Procedura `NewgetVisible` pro bod na pozici *PointNum* ve vektoru *Pts* určí viditelné body a jejich pozice v tomto vektoru uloží do vektoru *Reslt*. Teoreticky byl tento způsob již posán v kapitole 3.2.3. Funkce nastaví lokální proměnné a stav na „visible“. První cyklus prochází body napravo od zvoleného bodu směrem ke konci vektoru. Proměnná *test* nese výsledek skalárního součinu $-n \cdot p$, proměnná

cosangle součin $-n_1 \cdot p$. Poté nastává kontrola stavu současného viditelného bodu. V případě, že je predikován jako viditelný, je nejprve zkontrolováno, zda nesplňuje podmínku lokálního maxima vyššího, než zvolený bod. V případě, že tato platí, další stav je nastaven na „hidden“ a výška je uložena do proměnné *maxheight* pro hledání následujícího stejně vysokého bodu. Pokud tato podmínka neplatí, nastává testování výsledků skalárních součinů. Bod splňující obě podmínky je uložen do vektoru výsledků. Pokud ani jedna z předchozích situací nenastala, predikce dalšího bodu je nejistá, a tím pádem je zvolen stav „posshidden“.

Stav „hidden“ pouze čeká, až je nalezen bod alespoň stejně vysoký jako hodnota uložená v *maxheight*, aby přepnul predikci následujícího bodu na „posshidden“. Tento stav je pak implementačně blízký stavu „visible“, pouze v případě úspěšné kontroly skalárních součinů funkcí `SendRayTo(vector<points> Pts, int PointNum, int PointTgt)` kontroluje, zda lze vyslat ze zvoleného bodu *PointNum* do vyšetřovaného bodu *PointTgt* přímý paprsek. Tato funkce napřed zjišťuje směr vysílaného paprsku rozdílem X-ových složek počátečního a cílového bodu, v závislosti na tom se bude X-ový index bodu na paprsku zvětšovat nebo zmenšovat. Při kladném rozdílu, tedy pokud X-ová souřadnice zdroje je menší, než cíle, funkce projde všechny body ve vektoru mezi počátečním a cílovým bodem se stejnou X-ovou souřadnici (vzhledem k implementaci jich může být víc) jako má současný X-ový index bodu paprsku, a porovná Y-ovou souřadnici těchto bodů s Y-ovým indexem paprsku. Pokud bude index paprsku menší, než souřadnice bodu, paprsek se dostal pod zem, což znamená, že byla nalezena překážka, která brání těmto bodům se vidět, funkce vrátí negativní výsledek. Pokud byl nalezen bod s vyšší X-ovou souřadnicí, zvýší se rovněž X-ový index paprsku, takže se paprsek posune o jeden bod blíže k cíli, a proces se opakuje. Pokud za celý průchod nebylo nalezeno místo, kde by se paprsek dostal pod povrch, funkce vrací kladný výsledek. Vysílání paprsku směrem vlevo funguje obdobně, pouze X-ový index paprsku klesá.

Procedura `NewgetVisible` se taktéž ve výpočtu viditelných bodů nalevo a napravo od daného bodu liší pouze v tom, že doleva index klesá, a také vzhledem k tomu, že jde „proti směru“ směrnic povrchu, rozeznává klesání jako rostoucí směrnici. Tímto funkce `getAblation (vector<points> Pts, int PointNum, vector<points> Slopes)` získá seznam indexů bodů viditelných z bodu *PointNum* a dále už jen počítá pro tento bod rychlost klesání podle vzorců světelné odrazivosti a ablace . Tyto výpočty až na nutnou diskretizaci (převedení integrace na sumu) odpovídají vzorcům 2.11 a 2.1. Funkce `getHeight(vector<points> Pts,vector<points> Slopes)` takto v cyklu vytvoří změny pro všechny body. Nyní je potřeba pro body, které mají stejnou X-ovou souřadnici, sjednotit výsledek. Funkce prochází po hodnotách X-ové souřadnice a hledá body s touto souřadnicí ve vektoru *Pts*. Pokud je nalezen jediný bod s touto souřadnicí, je mu přidělena jeho rychlost klesání, v případě, že je takovýchto bodů víc, je rychlost klesání v tomto místě určena průměrem rychlostí bodů na tomto místě. Hledání bodů provádí funkce `getIndex(vector<points> Pts, int c)`, která vrací vektor obsahující všechny indexy bodů s X-ovou souřadnicí odpovídající hodnotě *c*. Funkce `setHeight(vector<double> input, vector<double> heights, double time)` prochází vytvořený vektor výšek a vektor výškové mapy a provádí krok simulace tak, že od každého bodu vektoru mapy odečte výšku, která na něm ubyla za čas daný parametrem.

Kapitola 5

Experimenty a vyhodnocení

V této kapitole jsou popsány experimenty, které byly nad modelem prováděny, zhodnocení jejich výsledků a to, jaký vliv měly na další práci na implementaci.

5.1 Popis experimentování

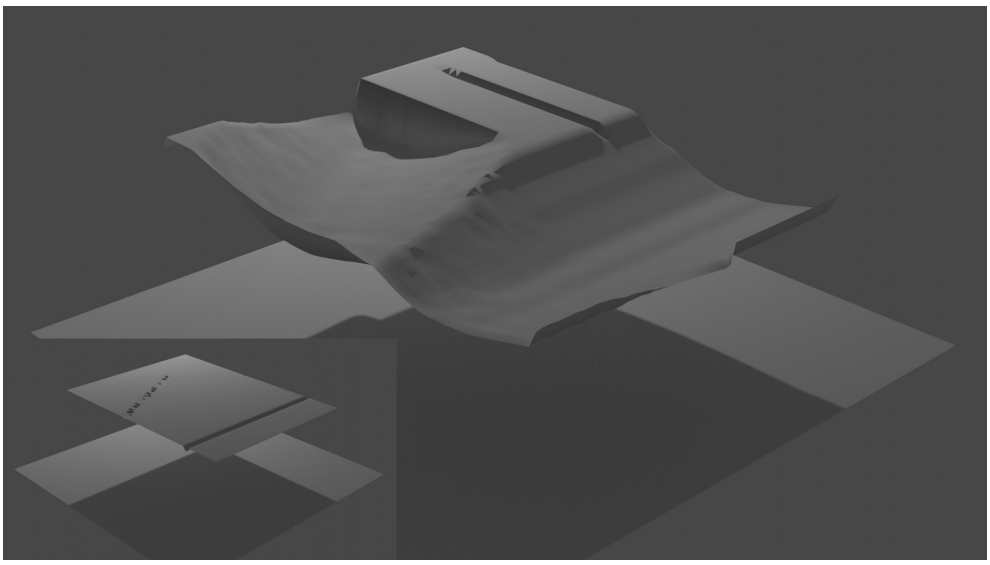
Na výsledném modelu byly prováděny experimenty, jejichž cílem bylo porovnat chování implementačního modelu s tvorbou kajícníků v reálném světě. K testování byl použitý notebook Lenovo B70 s čtyřjádrovým procesorem Intel i5-5200U 2,20 GHz a pamětí RAM 12 GB s grafickou kartou Intel HD Graphics 5500, a s nainstalovaným 64bitovým OS Windows 7 Professional.

Pro vysvětlení experimentů je důležité zde zmínit podmínky, které jsou v modelu nastaveny. Model nepočítá se změnou pozice Slunce na obloze, ani intenzity jeho záření a sluneční svit je nepřetržitý, nedochází proto ke střídání dne a noci. Těmito vlastnostmi se model podobá spíše umělým podmínkám experimentu Vance Bergerona, Charlese Bergera, a M. D. Bettertonové popsaného v kapitole 2.2, což byl také zřejmě výchozí bod pro matematický model, který M. D. Bettertonová následně vytvořila. Jelikož je tento matematický model, který byl zvolen k implementaci, popsán v centimetrech jako jednotkách délky, byla tato jednotka zvolena i jako jednotka délky tohoto modelu. V případě výškové mapy je tak délka strany 1 pixelu rovna 1 centimetru. Soubor .exr uchovává hodnoty barvy v datovém typu čísla s plovoucí řádkou, kde jsou viditelné hodnoty v intervalu hodnot <0,1>, což jsou pro simulaci v centimetrech nevhodné rozměry. Proto je výška bodů uvnitř programu škálována konstantou *resizer*. Tato konstanta má pro experimenty nastavenou hodnotu 300, tedy bod na mapě s hodnotou barvy 1 určuje výšku 3 metry. V případě, že je k tvorbě výškových map použitý Grafický editor GiMP 2.10, je třeba zmínit, že ten není prvotně připraven na práci s tímto typem souborů, a tak lze vkládat pouze hodnoty v rozmezí 0-255 pro jeden kanál. V případě kombinace těchto použitých faktorů vychází, že nejmenší zapsatelný výškový rozdíl mezi dvěma body v tomto programu je 1,18 cm. Adobe Photoshop je tak pro tvorbu výškových map vhodnější.

Prvotní testování probíhalo už během implementace nad jednotlivými vektory bodů, tedy řezy mapy. K tomu sloužila funkce `RunVectorWithVisual(vector<double> &v, int time)`. Tato funkce se stále nachází v souboru *functions.h*, i když už není samotným programem využívána. Má stejné rozhraní a princip, jako `RunVector(vector<double> &v, int time)`, přidává ovšem jednoduché grafické zobrazení řezu do příkazové řádky, kde tak lze sledovat body povrchu a jejich výšky po každém cyklu výpočtu. Tato funkce je implemen-

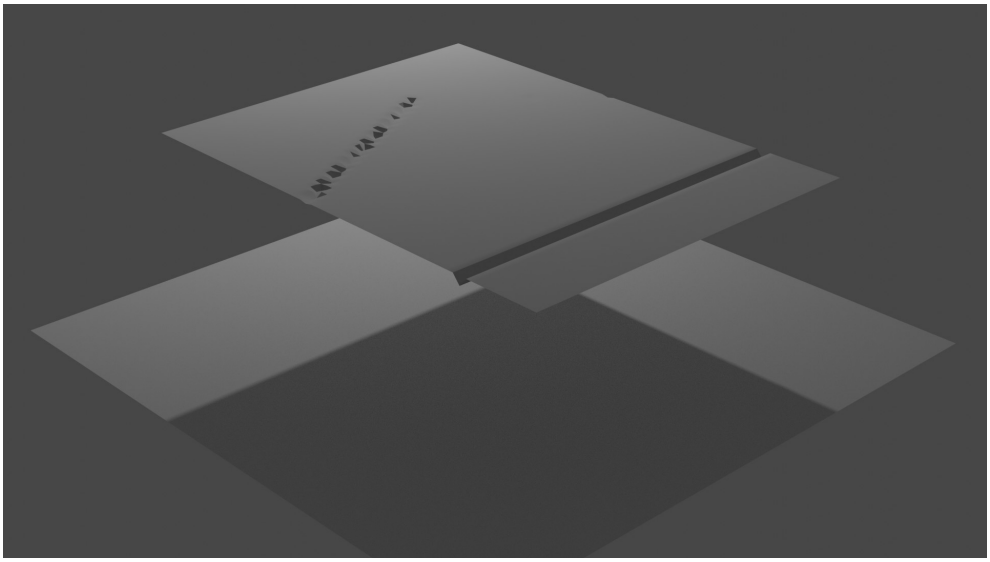
tována pomocí systémových volání Windows, proto není použitelná na nekompatibilních platformách, rovněž její použití v trojrozměrném prostoru by generovalo ohromné množství dat a zpomalovalo výpočet. Účel této funkce je pouze pro jednoduchou vizualizaci jednoho procházeného řezu určenou k ověřování správnosti algoritmu pro výpočet změny výšky v bodech řezu.

Experimentování na celém modelu počalo po implementaci všech potřebných součásti. Cílem těchto experimentů bylo zejména ověřit chování tohoto systému. K tomu byly v programu GiMP 2.10 připraveny výškové mapy obsahující počáteční terény, nad kterými poté program prováděl simulování průběhu vzniku kajícníků popsané modelem. Některé mapy obsahovaly přesně zadané body s nižší výškou oproti okolí, tyto byly použity zejména pro ověření očekávaných vlastností modelu, jiné mapy pak obsahovaly náhodné prohlubně generované šumem, ty napodobovaly vlastnosti reálné vrstvy sněhu, na které byl sledován vznik struktur kajícníků. Na základě výsledků těchto testů pak došlo k ladění výpočtů změny výšky povrchu a algoritmu obecně. Nejdůležitější informací získanou z první iterace testů bylo, že původní model M. D. Betterton je v ploše naprosto nedostačující. Tento model totiž počítá s jednotlivými řezy, které jsou ale při výpočtu nezávislé na svém okolí, což vede k nepřesným výsledkům. Jednak zdaleka nezapočítává do výpočtu viditelné body mimo řez, kterých může být v ploše mnoho, a rovněž nezávislost sousedních řezů může vést k obrovským rozdílům mezi sousedními body z různých řezů. Zejména nezávislost sousedních bodů vede k situaci, kdy se netvoří realistické struktury, protože se předpokládá, že šířka řezu je zanedbatelná. Příklady výsledků experimentů s tímto modelem zachycují obrázky 5.1 a 5.3.



Obrázek 5.1: Render výsledku experimentu po 576 krocích s dt=1h. Původní model zvládá vytvořit prohlubování, ale realistický terén vzniká pouze, pokud jsou sousední řezy nějak provázané. Model si poradil s prohlubněmi, které byly k řezu kolmé nebo pod úhlem, na kraji ale vytvořil nerealistickou svislou „zeď“, a jednobodový důlek se rovněž zvětšil pouze do délky.

Korektní model by měl na viditelnost vyšetřovat všechny body plochy. Zde ovšem přichází problém v tom, že matematický model tohoto chování by vyžadoval celou koncepci



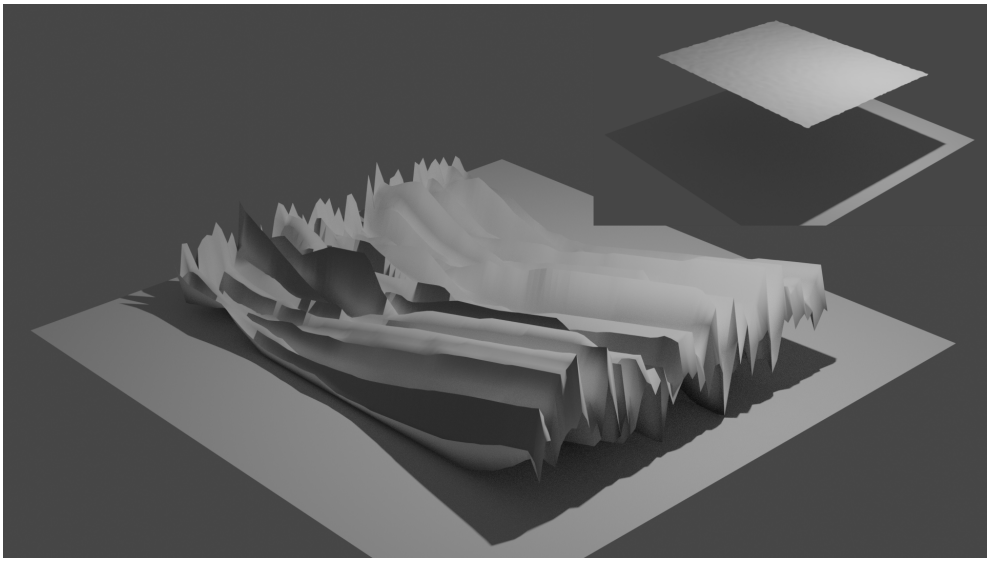
Obrázek 5.2: Render vstupního terénu pro výsledek zobrazený v 5.1 pro úplnost.

modelu předělat, protože zohlednění trojrozměrného prostoru by se dotklo i vzorců výpočtu, zejména problém určení viditelných bodů, konkrétně jejich přivrácenosti a detekce překážek mezi nimi, nebyl uspokojivě vyřešen. Řešení realizované v této práci spočívá v tom, že kromě řezů z řádků mapy probíhá výpočet následně i v řezech na ně kolmých, to znamená datech ve sloupcích výškové mapy. tím dochází k vytvoření vztahů mezi dosud nezávislými rovnoběžnými řezy. V jednom cyklu jsou nejprve vypočítány změny ve všech řádcích, ty jsou uloženy a následně nad změněnou plochou vypočteny změny ve sloupcích. Tím se ovlivňují body, které byly dosud navzájem nezávislé. Tento model tak vytváří terén, který působí značně realističtější, než původní model s nezávislými řezy. Dalším kladem tohoto řešení je i to, že je na viditelnost vůči jednomu bodu vyšetřováno více bodů plochy, než je tomu u původního modelu. Stále to ale nejsou všechny, jak by tomu mělo být teoreticky správně. Výsledky tohoto modelu zobrazují obrázky 5.5 a 5.9.

Časová náročnost výpočtu obou modelů závisí zejména na velikosti vstupní plochy, tedy počtu pixelů (bodů) ve výškové mapě. Pro soubor o rozměrech $200 \times 100px$ trvá výpočet 24 kroků simulace, přičemž dt=1 hodina, zhruba 62 sekund pro jednorozměrný model, vylepšený plošný model na stejný výpočet potřebuje 105 sekund. To může znamenat problém pro větší plochy, nicméně je zde poměrně velký potenciál v paralelizaci výpočtů, protože rovnoběžné řezy jsou na sobě nezávislé a je možné jejich výpočty provádět současně na více jádrech. Paralelizace kódu v místech výpočtu výšky bodů v řezu byla realizována s pomocí OpenMP. Paralelizovaný výpočet kroku při stejných parametrech trvá na čtyřjádrovém procesoru použitém k testování 35 sekund.

5.2 Vyhodnocení výsledků

Základní matematický model, který navrhla M. D. Betterton, pokrývá vlastnosti důležité pro vznik struktur známých jako kajícníci, ale jeho implementace je pro práci v prostoru nedostačující zejména pro neprovázanost sousedních řezů, která vede k vzniku nerealistického reliéfu. V něm mohou vznikat sousedící místa s extrémně velkými rozdíly ve výšce na

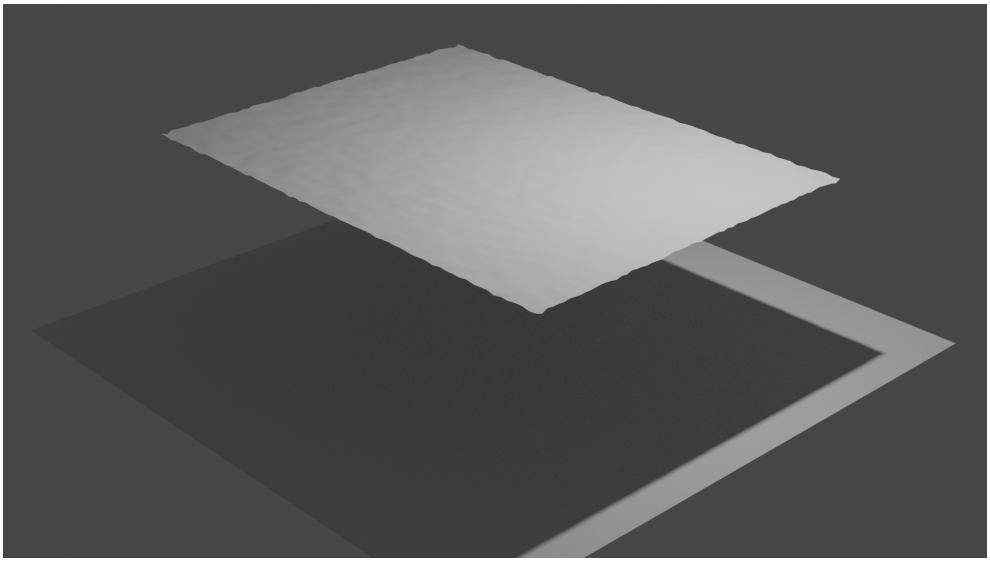


Obrázek 5.3: Render výsledku simulace nad terénem tvořeným plochou rozbitou šumem, tedy množstvím náhodných bodů s navzájem mírně odlišnou výškou.

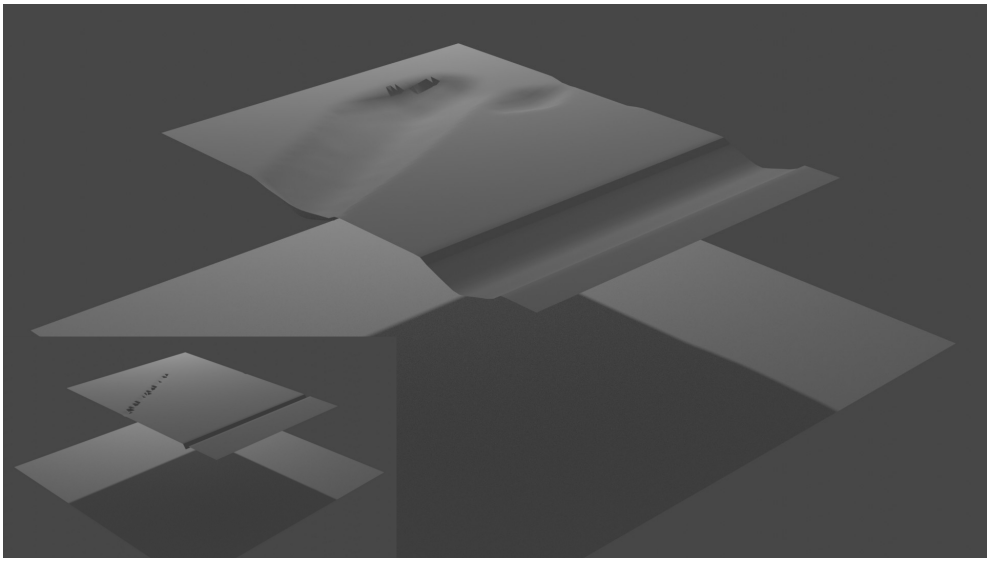
vzdálenosti sousedních řezů. Jelikož tato vzdálenost by pro vysokou přesnost měla být minimální, nelze tento model použít jako věrohodný. Pouhým přidáním dalšího průchodu řezy kolmými na původní řezy bylo dosaženo daleko věrohodnějšího vzhledu ledových útvarů. Během experimentů se podařilo ověřit očekávané vlastnosti tohoto modelu a rovněž to, že popsané chování skutečně vede ke vzniku struktur podobajícím se daným ledovcovým útvarům. Jevy stojící za vznikem těchto struktur jsou ale zřejmě ještě komplexnější a vyžadovaly by další práci nejen v modelování, ale i ve výzkumu těchto jevů ve skutečném systému polí kajícníků, jelikož jak již bylo zmíněno ne začátku této práce, tento jev stále není dokonale prozkoumán.

5.3 Možnosti budoucího vývoje

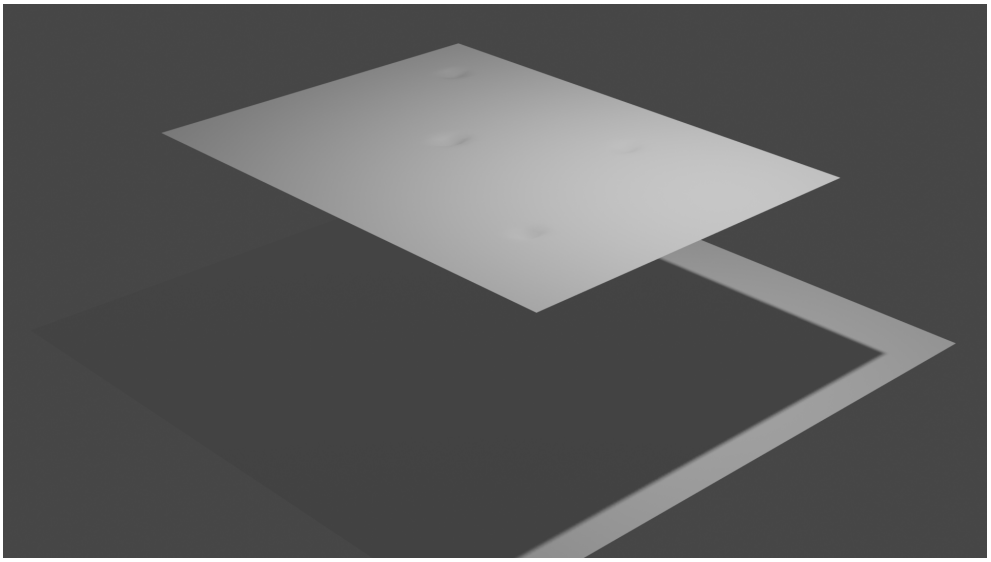
Tato práce se zabývala jedním matematickým modelem, který je oproti reálnému systému značně zjednodušen, například nezohledňuje vícenásobné odrazy světla, uvažuje jako konstantní fyzikální vlastnosti, které se ale v reálném systému mění v závislosti na vlastnostech okolí, a podobně. Rovněž se nejedná o plně plošný model, ale pouze jeho aproximaci pomocí využití řezů. Jako budoucí vývoj by tak mohlo být vhodné tyto vlastnosti do modelu zapracovat a porovnat výsledky tohoto rozšířeného modelu s tím, který vznikl v rámci této práce. Rovněž sporné vlivy, jako účinky větru nebyly dostatečně prozkoumány. Další možný a rozhodně zajímavý směr by mohl spočívat v simulaci chování modelu se zapracováním nečistot na povrchu, kterým se M. D. Betterton zabývá v druhé části své práce [7]. To by mohlo vést i k praktickému významu, jelikož vznik kajícníků na ledovci pomáhá k udržení vody uvnitř něj a umělé tvoření kajícníků pomocí prachu by tak mohlo pomoci zachovat ledovce [4], což by v současném stavu klimatu mohla být vítaná znalost.



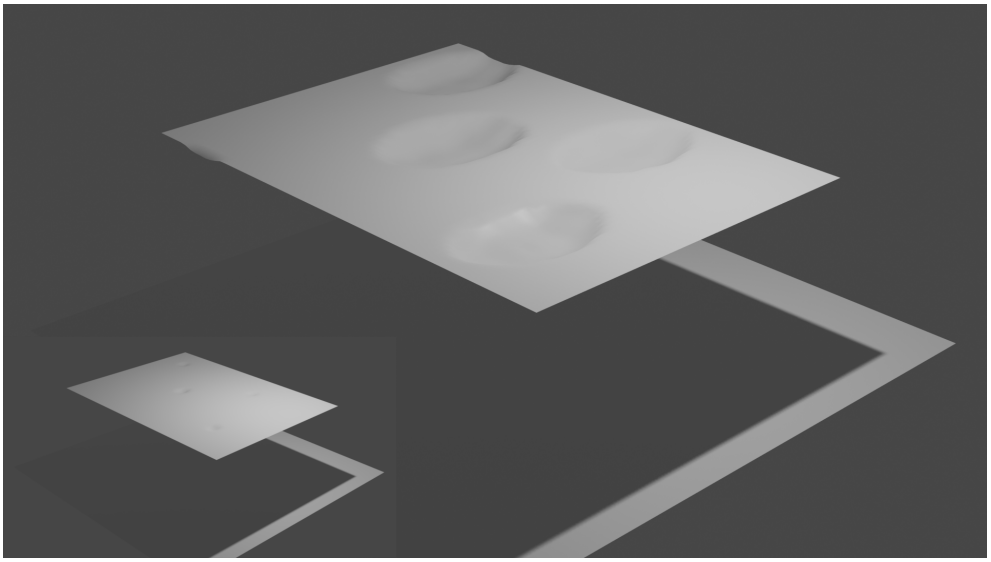
Obrázek 5.4: Render vstupního terénu pro výsledek zobrazený v 5.3.



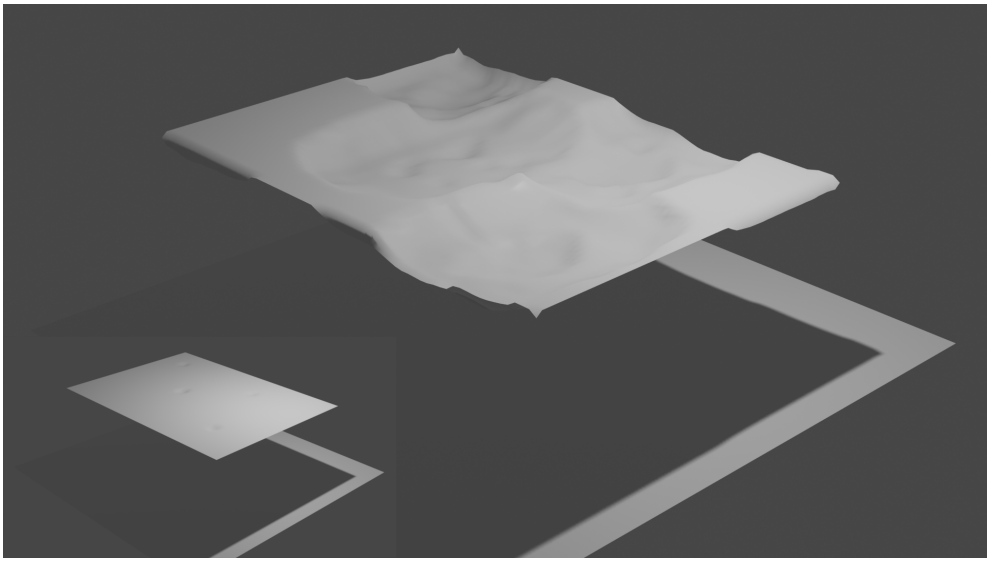
Obrázek 5.5: Render výsledku simulace nad terénem zobrazeným v 5.2. Oproti výsledku předchozího modelu na 5.1 stojí za povšimnutí daleko realističtější zakulacený okraj konce prohlubně diagonální k terénu místo „zdi“, který vytvořil původní model. Rovněž i důlek o velikosti 1 pixel vytvořil oblou prohlubeň daleko více odpovídající reálnému předpokladu.



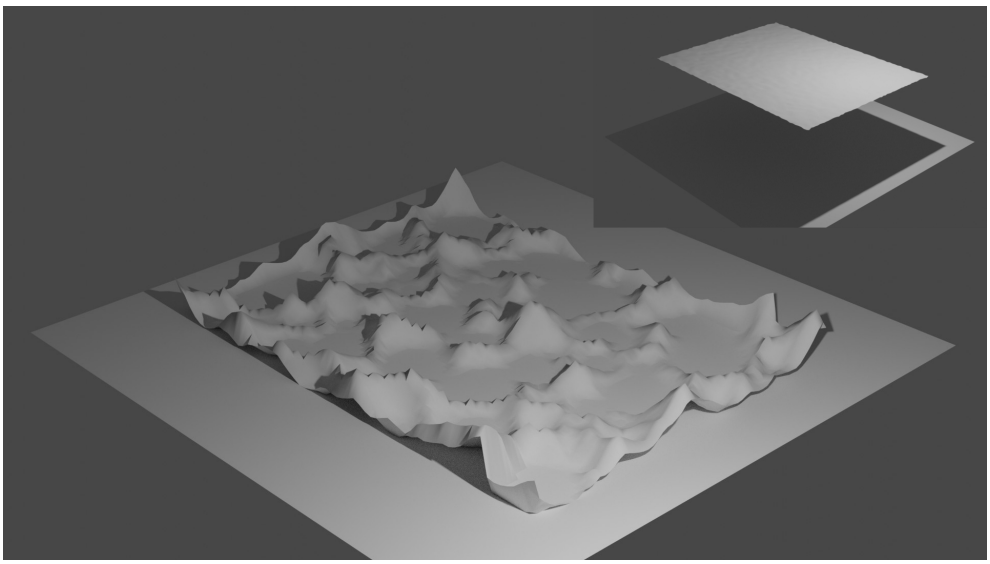
Obrázek 5.6: Tento experiment ukazuje základní vlastnosti modelu. Toto je počáteční stav ledové vrstvy, s čtyřmi počátečními prohlubněmi pro zahájení ablace.



Obrázek 5.7: Terén z obrázku 5.6 po 96 hodinách simulace, prohlubně se rozšiřují do všech tří os, zatímco rovný terén se nemění.



Obrázek 5.8: Po dalších 96 hodinách simulace od situace zobrazené na 5.7 stěny mezi jednotlivými prohlubněmi začínají tát a snižovat se. To vede k zvětšení viditelnosti mezi body v různých prohlubních. Vznikají další místa, která se rychleji prohlubují, stěny prohlubní, do kterých se odráží světlo z jiných prohlubní také tají rychleji.



Obrázek 5.9: Render výsledku simulace nad terénem zobrazeným v 5.4 po 576 krocích s dt=1h. Opět stojí za pozornost fakt, že terén působí pravděpodobně. Zde lze pozorovat struktury odpovídající kajícníkům.

Kapitola 6

Závěr

V rámci této bakalářské práce jsem se seznámil s problematikou týkající se vzniku ledovcových struktur známých jako Kajícníci a nastudoval literaturu, která tuto problematiku rozvíjí. Tyto získané informace následně posloužily pro vysvětlení problematiky v první kapitole této práce. Na základě toho pak došlo k vybrání vhodných technologií, pomocí kterých byla práce realizována, a rovněž byl vytvořen návrh řešení praktické části projektu na základě jednodimenzionálního modelu od M. D. Betterton. Současně s návrhem byla připravena i následná implementace modelu v jazyce C++, který napodobuje tvorbu ledovcových útvarů známých jako kajícníci. Tato implementace je taktéž v práci popsána.

Nad tímto modelem byly poté prováděny experimenty, které měly ověřit jeho věrohodnost. Původní jednodimenzionální model byl vyhodnocen jako nedostatečně věrohodný a proto byla na jejich základě vytvořena úprava původního jednorozměrného modelu spočívající v průchodu na sebe kolmých řezů, reprezentujících řádky a sloupce výškové mapy. Tato jednoduchá úprava značně přibližuje výsledky modelu realitě.

Během práce byly taktéž vytvořeny následující materiály: video, plakát a článek, který byl následně zařazen do studentské konference Excel@FIT 2020. Tyto materiály jsou přiloženy na disku.

Literatura

[1] Penitents in Argentine, near Aconcagua.
URL https://commons.wikimedia.org/wiki/File:Penitentes-Aconcagua.jpg

[2] Vztažná a souřadnicová soustava - Bakalářská fyzika pro HGF VŠB-TUO.
URL http://if.vsb.cz/bf/04.html

[3] Anderson, P. S.: Europa may have towering ice spikes on its surface.
URL
https://earthsky.org/space/jupiters-moon-europa-penitentes-ice-spikes

[4] Ball, P.: Could a sprinkling of dirt save the glaciers?
URL https://www.nature.com/news/2006/060130/full/060130-12.html

[5] Barrrera, H. V.: *A study of the Nieve Penitente of the Chilean Andes*. Union Internationale de Géodésie et de Géophysique, Association Internationale d'Hydrologie Scientifique. Assemblée Générale à Erlimbourg, 1936, translation by B.F. Chappelle.

[6] Bergeron, V.; Berger, C.; Betterton, M. D.: *Controlled Irradiative Formation of Penitentes*. 2006.

[7] Betterton, M. D.: *Formation of Structure in Snowfields:Penitentes, Suncups, and Dirt Cones*. Physical Review E, vol 63, 2001.

[8] Brahic, C.: Spiky glaciers are slower to melt.
URL https://www.newscientist.com/article/dn11309-spiky-glaciers-are-slower-to-melt/

[9] Corripio, J. G.: *Modelling the energy balance of high altitude glacierised basins in the Central Andes*. The University of Edinburgh, 2002.

[10] Darwin, C.: *The Voyage of the Beagle*. 1839.

[11] Fortmann, M.: The Holy Week in Granada.
URL
https://www.escuela-montalban.com/blog-and-news/blogdetails.php?newsID=8376&cat=4

[12] Hoffmann, T.: SunCalc.
URL https://www.suncalc.org/#/-22.106,-70.3125,2/2019.12.26/14:23/1/3

[13] Lliboitry, L.: *The Origin of Penitents*. Journal of Glaciology, s. 2:331–338, 1954.

[14] Swithinbank, C.: *The Origin of Dirt Cones on Glaciers*. Journal of Glaciology, Volume 1, Issue 8, Oxford University Exploration Club, 1950.

[15] Takahashi, S.; Fujii, T.; Ishida, T.: *Origin development of polygonal ablation hollows on a snow surface*. Low Temperature Science Series A, (31):191, 1978.

[16] Villeneuve, S.: Obzorníkové souřadnice, Wikipedie.
URL https://cs.wikipedia.org/wiki/Obzorníkové_souřadnice