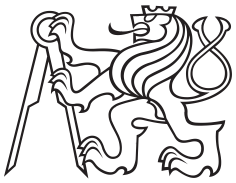
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Simulace lavin

Bc. David Vyvlečka

Vedoucí: Ing. Jaroslav Sloup
Květen 2017

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Vyvlečka** Jméno: **David** Osobní číslo: **406988**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**

Studijní program: **Otevřená informatika**

Studijní obor: **Počítačová grafika a interakce**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Simulace lavin

Název diplomové práce anglicky:

Simulation of mixed-motion avalanches

Pokyny pro vypracování:

Seznamte se s metodou simulace lavin popsanou v [1], která využívá kombinaci částicového systému (Smoothed Particle Hydrodynamics) a mřížkové metody pro výpočet pohybu sněhu a jeho interakce s okolím. Na základě prostudované literatury navrhněte a implementujte aplikaci, která bude simulovat šíření laviny ve scéně popsané pomocí výškové mapy. Součástí aplikace bude přehledné uživatelské rozhraní umožňující měnit jednotlivé simulační parametry (např. velikost mřížky, počet částic, hustotu sněhu, ...). Vizualizace bude řešena minimalistickým způsobem tak, aby dostatečně průkazně prezentovala výsledky simulace. Funkčnost implementované metody ověřte alespoň na třech scénách různé komplexnosti a výsledky porovnejte s videi skutečných lavin. Implementovanou metodu zhodnoťte z hlediska rychlosti a analyzujte časovou náročnost jednotlivých částí simulace.
Implementaci proveďte v C/C++ s využitím OpenGL a technologie CUDA.

Seznam doporučené literatury:

[1] Yusuke Tsuda, Yonghao Yue, Yoshinori Dobashi, Tomoyuki Nishita: Visual simulation of mixed-motion avalanches with interactions between snow layers. The Visual Computer, Vol.26, Issue 6-8, pp. 883-891, Springer-Verlag, 2010.
[2] Tetsuya Takahashi, Issei Fujishiro, Tomoyuki Nishita : Visual Simulation of Compressible Snow with Friction and Cohesion. NICOGRAPH International 2014, pp.35-42.
[3] D. Gucer, B. Ozguc: Simulation of a flowing snow avalanche using molecular dynamics. Turkish Journal of Electrical Engineering & Computer Sciences. Vol.22, No.6, p. 1596-1610, 2014.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Jaroslav Sloup, Katedra počítačové grafiky a interakce

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **24.01.2017** Termín odevzdání diplomové práce: **26.05.2017**

Platnost zadání diplomové práce: _____

Podpis vedoucí(ho) práce	Podpis vedoucí(ho) ústavu/katedry	Podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____ Datum převzetí zadání

_____ Podpis studenta

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Jaroslavu Sloupovi za pravidelné konzultace a hodnotné náměty a rady při vypracování této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 23. května 2017

Abstrakt

Tato diplomová práce se věnuje návrhu a implementaci aplikace simulující chování lavin kombinovaného typu v terénu popsaném výškovou mapou. Návrh vychází z průzkumu existujících řešení daného problému. Výsledná aplikace využívá kombinace částicového systému SPH a mřížkové metody k simulaci různých vrstev laviny. Jádro aplikace je implementováno v jazyce C++ za podpory paralelního zpracování v technologii CUDA. Uživatelské rozhraní je postaveno na knihovně Qt a vizualizace je řešena za pomoci knihovny OpenGL. Implementovaná aplikace umožňuje uživateli simulovat šíření laviny v libovolném terénu a v závislosti na nastavených parametrech simulace.

Klíčová slova: lavina, simulace, SPH, sníh, C++, CUDA, OpenGL

Vedoucí: Ing. Jaroslav Sloup

Abstract

This master's thesis is devoted to designing and implementation of application simulating mixed-motion avalanches in terrain described by heightmap. Design is based on a survey on existing solutions to this problem. The resulting application uses a combination of SPH particle system and grid method to simulate different avalanche layers. The core of the application is implemented in C++ with support for parallel processing in CUDA. The user interface is built on the Qt library and visualization is done using the OpenGL library. The implemented application allows the user to simulate the propagation of the avalanche in any terrain depending on the simulation parameters set.

Keywords: avalanche, simulation, SPH, snow, C++, CUDA, OpenGL

Title translation: Simulation of mixed-motion avalanches

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Laviny představují katastrofální přírodní jev, jehož studium a následná simulace může přispět k predikci jeho výskytu, rozměrů a především k včasnému varování před hrozícím nebezpečím a omezení ztrát na životech tímto jevem způsobených.

Tato práce si klade za cíl na základě prostudované literatury o tvorbě lavin a existujících řešeních jejich simulace navrhnout aplikaci, která umožní simulovat šíření laviny v terénu popsaném pomocí výškové mapy. Jelikož se práce zaměřuje na samotnou simulaci, je výstup prezentován pouze jednoduchou vizualizací. Stěžejním bodem zájmu jsou laviny tzv. kombinovaného typu, viz kapitola 2, které obvykle dosahují největších rozměrů a největší ničivé síly.

Obrázek 1.1: Ukázka laviny padající z horského úbočí, kde je vidět zároveň její hlavní část tvořená hustějším sněhem i oblak tvořený sněhem prachovým. Převzato z [3].

Práce je dále členěna následovně: v kapitole 2 je poskytnut přehled dělení lavin na jednotlivé typy a nastíněny základní fyzikální principy stojící za jejich vznikem. S těmito znalostmi byla zpracována rešerše existujících metod pro simulaci lavin různých typů na počítači, která je poskytnuta v kapitole 3. Uvedené metody poskytují ucelený přehled, z nějž vychází kapitola 4. V této kapitole je uveden návrh aplikace, která je hlavním výsledkem této práce. Kapitola 5 následně podává podrobnosti týkající se jejího vzniku a nakonec v kapitole 6 jsou podány výsledky, kterých bylo v rámci práce dosaženo.

Kapitola 2

Laviny

Laviny jsou rychlé gravitací řízené masy sněhu pohybující se směrem dolů po horském svahu. Tato základní definice uvedená v [2] odlišuje laviny od dalších pohybů sněhové pokrývky, jako např. závějí způsobených větrem nebo od pomalých sesuvů, které jsou sice způsobeny gravitací, ale nedosahují rychlosti lavin. Práce [2] zároveň uvádí základní dělení lavin na dva typy dle převažujícího pohybu:

- Tekoucí lavina (flowing avalanche nebo také tzv. dense-flow avalanche [4]) je lavina s jádrem tvořeným sněhem o vysoké hustotě, viz obrázek 2.1a.
- Vzdušná lavina (airborne avalanche) je velmi rychle se pohybující oblak sněhu, viz obrázek 2.1b.

(a): Tekoucí lavina bez přítomnosti oblaku tvořeného prachovým sněhem, převzato z [3].

(b): Lavina tvořená prachovým sněhem. Sníh je vynesen vysoko nad terén. Převzato z [4].

Obrázek 2.1: Porovnání lavin z hlediska převažujícího typu sněhu, laviny tvořené sněhem o vysoké hustotě a laviny tvořené prachovým sněhem.

Průměrné charakteristiky tekoucích i vzdušných lavin jsou uvedeny v tabulce 2.1. Některé další zdroje [4, 5] pracují i s termínem lavin kombinovaného

pohybu (mixed-motion avalanches), které vykazují charakteristiky obou výše uvedených typů. [2] však upozorňuje, že by termín měl být používán jen ve specifických případech, kdy má jak tekoucí, tak vzdušná část vliv na celkovou dynamiku laviny, a nikoliv v případech, kdy je tekoucí lavina přikryta oblakem sněhu vytlačeným z jádra působením tření vzduchu.

	Tekoucí lavina	Vzdušná lavina
Hloubka toku	$\sim m$	$10 - 100\ m$
Průměrná hustota	$150 - 500\ kg/m^3$	$5 - 50\ kg/m^3$
Průměrná rychlost	$5 - 25\ m/s$	$50 - 100\ m/s$

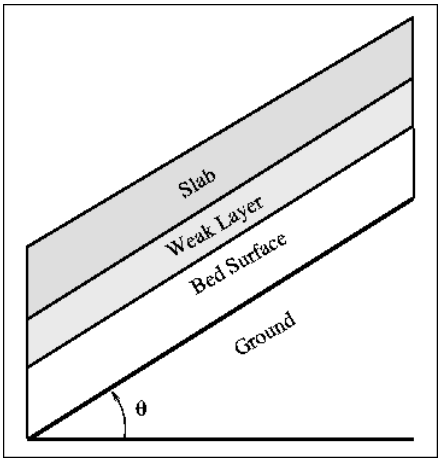
Tabulka 2.1: Charakteristika tekoucích a vzdušných lavin dle [2].

Jiné zdroje nabízí odlišná dělení lavin, např. [6] uvádí rozdělení na laviny tvořené sypkým sněhem nebo hroudami sněhu (loose snow avalanches), viz obrázek 2.2 a na deskové laviny (slab avalanches), viz obrázek 2.5. Tento typ dělení lavin dle vlastností sněhu, kterým jsou tvořeny, je mnohdy dále rozšiřován, např. na laviny mokré [7]. Mokré laviny, jak již název napovídá, jsou tvořeny mokrým sněhem a společně s lavinami tvořenými sypkým sněhem se šíří z jednoho bodu (na počátku se pohybuje malé množství sněhu) a získávají na velikosti strháváním sněhu v dráze pohybu. Laviny tvořené sypkým sněhem se pak nejčastěji vyskytují na svazích s velkým sklonem a tvoří typické, protáhlé trojúhelníkové formace [6].

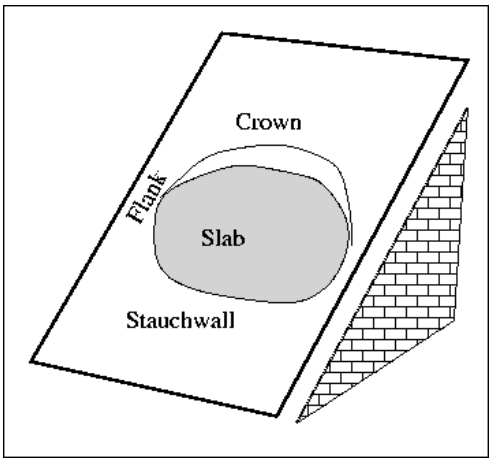
Obrázek 2.2: Ukázka laviny tvořené sypkým sněhem (loose snow avalanche). Je zřetelně vidět dva body v prudkém svahu, ze kterých se lavina rozšířila. Vzniklá formace tvarem připomíná trojúhelník. Převzato z [3].

Naopak deskové laviny jsou tvořeny soudržnou vrstvou sněhu, která se po svahu pohybuje jako jeden celek a až v důsledku kolizí s terénem a

dalšími objekty se láme na menší kusy [6]. Jsou to právě deskové laviny, které představují největší hrozbu při pohybu v horách a jsou zodpovědné za nejvyšší počet obětí [8] a je jim proto věnována velká pozornost.



(a): Komponenty deskové laviny.



(b): Kritická místa deskové laviny.

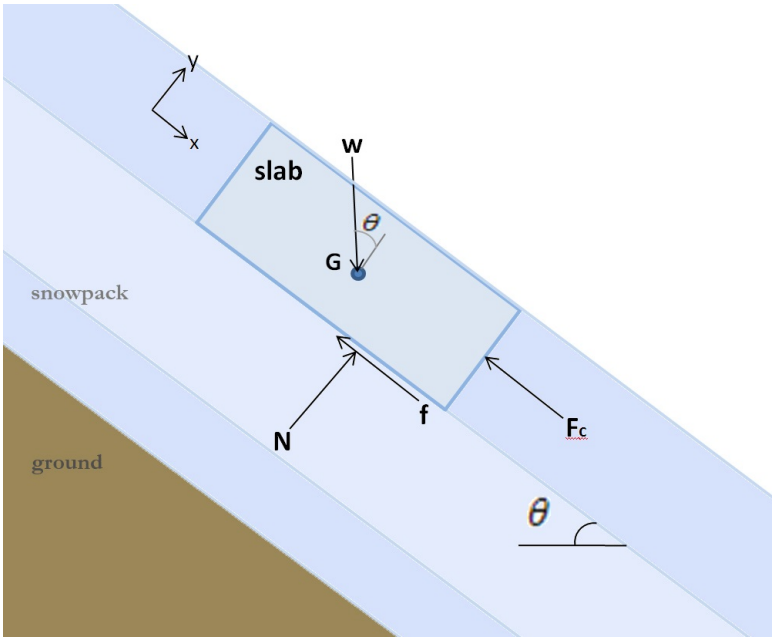
Obrázek 2.3: Anatomie deskových lavin, převzato z [9]. Na obrázku 2.3a jsou uvedeny komponenty deskové laviny: deska (slab), která se pohybuje po selhání slabé vrstvy (weak layer) po podloží (může být buď starší pevná vrstva sněhu nebo přímo terén). Na obrázku 2.3b jsou uvedena kritická místa deskové laviny: koruna (crown), kde dochází k namáhání v tahu, boky laviny (flanks), kde dochází ke smykovému namáhání a namáhání v tahu, deska (slab), která představuje pohybující se těleso laviny a spodní část laviny (stauchwall), kde působí namáhání v tlaku.

K uvolnění deskové laviny dochází v momentě, kdy síly, kterými působí deska, převáží nad silami, které udržují slabou vrstvu, tvořenou volným nesoudržným sněhem, pohromadě. V tomto momentě dochází k selhání slabé vrstvy a síly působící na zbylé oblasti propojující lavinu s terénem se rapidně zvyšují. Nejvyšší namáhání působí na korunu a boky laviny (viz obrázek 2.3b), vazby v těchto místech již obvykle nejsou dostatečně silné, aby udržely desku na místě a tak dojde k jejímu odtržení. Na spodní hranici laviny, kde nedošlo k selhání slabé vrstvy nebo se tato vrstva již ve sněhové pokrývce nevyskytuje, začínají s pohybem desky růst tlakové síly, které vyzdvihnou desku nad pokrývku.

Dle [6] je vzhledem k množství proměnných značně obtížné analyzovat tyto síly působící na lavinu jako celek. Pokud se ale omezíme na malou část desky, je možné situaci zjednodušit dle silového diagramu na obrázku 2.4.

Dle tohoto diagramu na blok desky s těžištěm v bodě $\mathbf{G}$ působí tíhová síla $\mathbf{w}$, třecí síla $\mathbf{f}$, tlaková síla $\mathbf{F_c}$ a normálová síla $\mathbf{N}$. Za předpokladu, že sklon svahu pod blokem odpovídá úhlu θ, pak k odtržení dojde v momentě, kdy je narušen rovnovážný stav (zrychlení ve směru souřadnice x je kladné) a je splněna následující nerovnice:

$$\mathbf{w} \sin\theta > \mathbf{F_c} + \mathbf{f} \tag{2.1}$$



Obrázek 2.4: Silový diagram izolované části deskové laviny, převzato z [6].

Uvedený vztah indukuje vliv hmotnosti sněhu na pravěpodobnost pádu laviny. Je zřejmé, že síly, které musí působit proti desce, má-li zůstat zachován rovnovážný stav, musí být vyšší v případě desky tvořené těžkým mokrým sněhem o vysoké hustotě, než síly působící na desku tvořenou sněhem suchým o hustotě nižší. Obdobný vliv by měl mít i narůstající sklon terénu; praktická měření [2] ukazují, že průměrný sklon svahu v místě odtržení laviny se pohybuje od 27^o do 50^o a pouze výjimečně dochází k lavinám na svazích se sklonem menším než 25^o. Naopak na svazích jejichž sklon přesahuje uvedené hodnoty nedochází k výraznému hromadění sněhu (již v průběhu tvorby sněhové pokrývky dochází k malým sesuvům) a riziko odtržení laviny klesá.

Kromě přítomnosti slabé vrstvy a kritického sklonu svahu existují zároveň další faktory ovlivňující tvorbu lavin [10]:

- **Nový sníh** – množství nového sněhu a rychlost jeho přibývání mohou ovlivnit slabou vrstvu pod ním. Pokud je množství srážek vysoké a nová sněhová pokrývka se tvoří velmi rychle, nemusí dojít k dostatečnému zpevnění slabé vrstvy, která zůstává nesoudržná.

- **Vítr** – vítr přispívá zejména k nerovnoměrnému hromadění sněhu v různých úsecích svahu a zvyšuje tak napětí působící mezi jednotlivými vrstvami. Rozdíly v rychlosti větru mohou být příčinou vzniku oblastí s rozdílnou hustotou a tvrdostí sněhové pokrývky.

- **Teplota** – Zvýšení vzdušné teploty, zejména pokud je velmi rychlé, také přispívá k nestabilitě sněhové pokrývky. Vzhledem k nízké tepelné vodivosti sněhu má teplota vliv primárně na vrchní vrstvu pokrývky.

Obrázek 2.5: Následky pádu deskové laviny v Modrém dole, Krkonoše, 10. února 2015. V levé části snímku je zřetelně vidět odtrhová hrana koruny laviny.

Tání může přispět ke změně tuhosti a odolnosti namáhání ve smyku, stejně jako k rychlejší propagaci trhlin.

2.1 Laviny z pohledu fluidní dynamiky

Z podstaty chování lavin vyplývá, že vykazují vlastnosti tečení, ať již kapalin (tekoucí laviny) nebo plynů (laviny tvořené prachovým sněhem). Tohoto jevu je bohatě využíváno i při simulaci lavin (viz následující kapitola), kdy mnoho metod vychází z fluidní dynamiky. Tok v kapalině je pak možno zachytit pomocí Navier-Stokesových rovnic [14]:

$$\frac{\delta \mathbf{u}}{\delta t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}, \tag{2.2}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{2.3}$$

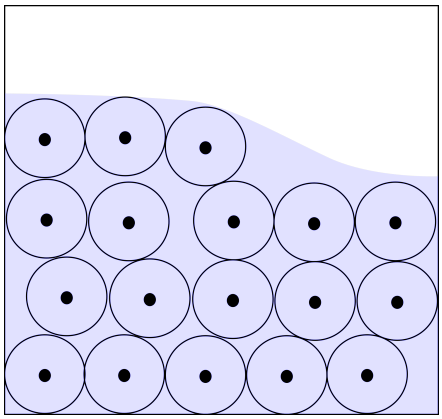
které popisují změnu rychlosti $\mathbf{u}$ a tlaku p v kapalině o téměř konstantní hustotě a teplotě za časový krok δt. Dále ν představuje viskozitu popisované kapaliny, ρ její hustotu. Rovnice 2.2 je rozdělena na jednotlivé členy, kde každý popisuje jistou část pohybu kapaliny. Člen $-(\mathbf{u} \cdot \nabla)\mathbf{u}$ popisuje proudění kapaliny a je zodpovědný za přenos kinetické energie. Člen $-\frac{1}{\rho}\nabla p$ vyjadřuje závislost na tlaku, kdy má kapalina tendenci šířit se do míst, kde je hodnota tlaku nižší. Zároveň garantuje nestlačitelnost kapaliny. Difuzní člen $\nu \nabla^2 \mathbf{u}$ vyjadřuje závislost na viskozitě kapaliny a nakonec $\mathbf{f}$ představuje externí síly působící na kapalinu. Rovnice 2.3 pak vynucuje zachovávání hmoty a rovnice 2.2 zachování kinetické energie v kapalině.

Analytické řešení Navier-Stokesových rovnic je možné jen v omezeném množství případů, při řešení praktických problémů se proto přistupuje k výpočtu numerickému. Metodami, které řešení Navier-Stokesových rovnic aproximují, jsou např. Lattice Boltzmann Method (LBM) [11], Coupled map lattice (CML) [12] rozšiřující metody buněčných automatů, dále pak metody Lagrangeovské, mezi něž patří např. molekulární dynamika [17] nebo Smoothed particle hydrodynamics (SPH) [18] a metody Eulerovské (mřížková metoda) [14]. Rozdíl mezi Lagrangeovskými a Eulerovskými metodami je uveden v následující kapitole, kde je zároveň představeno několik vybraných metod v kontextu simulace lavin.

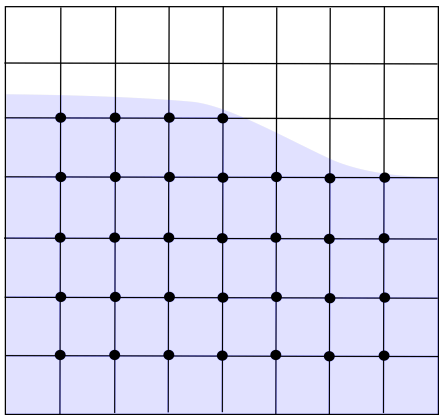
Kapitola 3

Simulace lavin

Z předcházející kapitoly vyplývá, že laviny jsou jevem velmi komplexním a jejich simulace zahrnující veškeré fyzikální faktory by byla přinejmenším velmi náročná. V praxi se proto používají metody, které tento problém zjednodušují. Mnohé z těchto metod jsou založeny na fluidní dynamice a jsou obvykle používány pro simulaci kapalin. Vzhledem k vzájemné podobnosti v chování sněhu v lavině s tečením je však možné dosáhnout výsledků, které věrně aproximují reálné laviny. V této kapitole jsou proto představeny metody doposud používané pro simulaci lavin nebo s lavinami související.



(a): Ukázka Lagrangeovské metody. Simulovaná látka je přímo vymezena pozicí částic, které se v čase pohybují.



(b): Ukázka Eulerovské metody. Látka je simulována pomocí mřížky nepohyblivých bodů. V čase se mění hodnoty uložené v bodech mřížky.

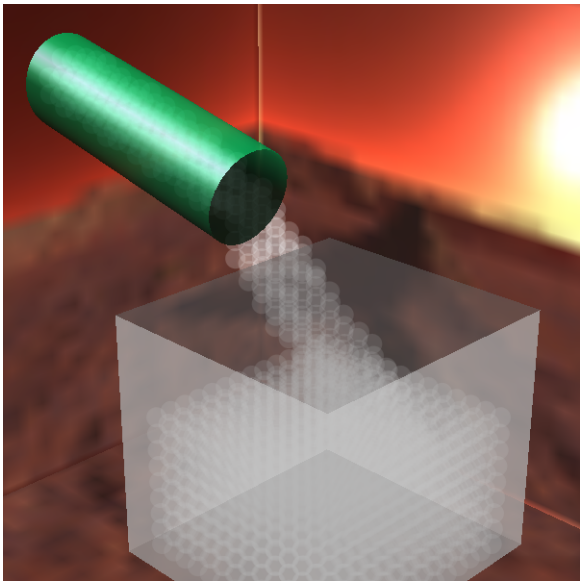
Obrázek 3.1: Porovnání dvou základních přístupů k simulaci kapaliny: Lagrangeovského a Eulerovského.

Všechny zde uvedené metody mají společný základ, snaží se aproximovat Navier-Stokesovy rovnice (2.3, 2.2), ale způsob, jakým je aproximace dosaženo se může značně lišit. V principu můžeme rozlišit dva základní přístupy k aproximaci toku v kapalině, základní rozdíl je naznačen na obrázku 3.1. První přístup využívá částic, nesoucích informaci o kapalině, které se přímo pohybují v závislosti na silovém působení, viz obrázek 3.1a. Takový systém,

který využívá pohybujících se částic, nazýváme Lagrangeovský, z uvedených metod mu odpovídá např. SPH [18] nebo molekulární dynamika [17]. Druhý přístup využívá diskrétního rozdělení prostoru fixními body, viz obrázek 3.1b, mezi kterými dochází k výměně informací o kapalině. Tento systém nazýváme Newtonovský, z uvedených metod mu odpovídá mřížková metoda [14]. Každý přístup má své vlastní výhody, v poslední době se proto objevuje i množství metod, které se snaží výhody obou přístupů zkombinovat. Tyto metody jsou uvedeny v samostatné sekci 3.3. Veškeré dále uvedené metody jsou již uvažovány v kontextu simulace lavin nebo sněhu, komplexnější shrnutí metod používaných ve fluidní dynamice nabízí např. [13].

3.1 Mřížková metoda

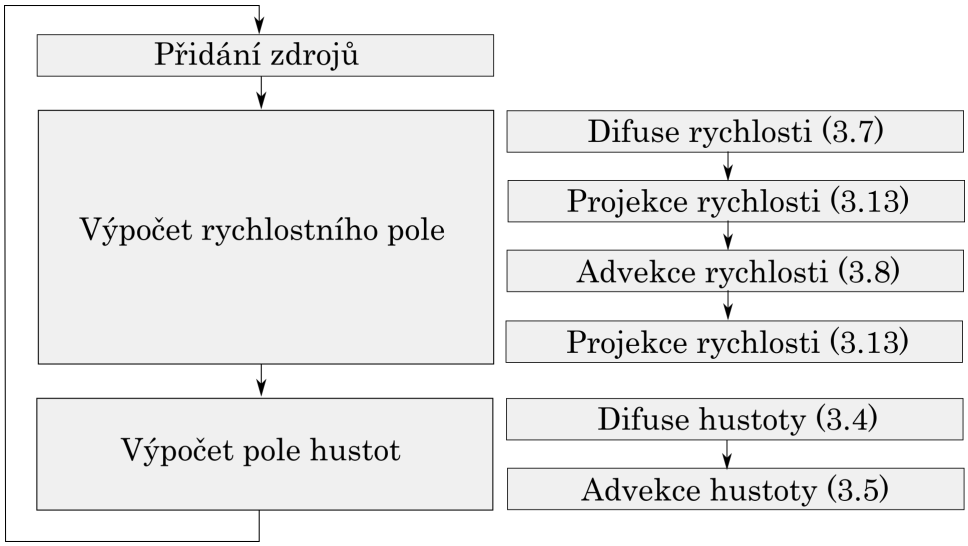
Mřížková metoda je jednou ze základních metod simulace kapalin a plynů, své využití nachází ale i při simulaci lavin. V práci [5] je mřížkové metody využito při simulaci laviny kombinovaného typu. Vzhledem ke svým vlastnostem je zde mřížková metoda využita při simulaci vrstvy tvořené prachovým sněhem. Tato kombinovaná metoda je blíže představena v samostatné sekci 3.3.



Obrázek 3.2: Simulace proudění kapaliny pomocí mřížkové metody, převzato z [15].

Samotná mřížková metoda [28] je založena na diskrétním rozdělení prostoru, ve kterém jednotlivé fixní body mřížky nesou informace o simulované kapalině nebo plynu. Nesenými informacemi jsou obvykle hustota a rychlost, ale mohou se zde vyskytovat i další (např. teplota). V kontextu této práce je dále uvažována pouze rychlost a hustota. Mřížka tedy představuje jak skalární pole (v případě hustoty) i pole vektorové (v případě rychlosti). Tato pole se na počátku simulace nechází v iniciální konfiguraci $a_0(x) = a(x,t)$ a $\mathbf{u}_0(x) = \mathbf{u}(x,t)$, přičemž $t = 0$ (a představuje skalární hodnotu, zde

hustota, **u** představuje rychlost, x označuje buňku mřížky a t počáteční čas), na jejímž základě simulace probíhá. Výsledek jednoho simulačního kroku $\mathbf{w}(x)$ (zahrnující jak vektorové, tak skalární pole) pak rovnou vstupuje do simulačního kroku dalšího. Jeden simulační krok pak sestává ze tří základních kroků dále rozdělených na podkroky, jak je uvedeno na schématu 3.3: přidání zdrojů, výpočet rychlostního pole a výpočet pole hustot.



Obrázek 3.3: Průběh mřížkové metody s rozdělením na podkroky. Projekce se ve výpočtu uplatňuje dvakrát, neboť advekční krok podává lepší výsledky, pokud pole zachovává hmotu [28]. U jednotlivých kroků je uvedena příslušná rovnice.

3.1.1 Přidání zdrojů

Simulace se odvíjí od počáteční konfigurace $\mathbf{w}_0(x)$, ale i v jejím průběhu může být ovlivňována novými vstupujícími hodnotami. Takové zdroje si můžeme představit např. jako komín, z nějž vychází kouř (zdroj hustoty) a vítr, který kouř rozfoukává (zdroj rychlosti). Přidání zdrojů probíhá dle jednoduchého schématu. Ke konfiguraci jsou přidány příslušné zdroje ve formě externí síly **f** nebo množství hustoty D dle velikosti časového kroku simulace:

$$\mathbf{u}_{added} = \mathbf{u}_{not\_added} + \Delta t \mathbf{f}(x,t) \tag{3.1}$$

$$a_{added} = a_{not\_added} + \Delta t D(x,t) \tag{3.2}$$

3.1.2 Výpočet pole hustot

Vývoj skalárního pole může být popsán advekčně-difuzní rovnicí, která má velmi podobnou formu jako první z Navier-Stokesových rovnic:

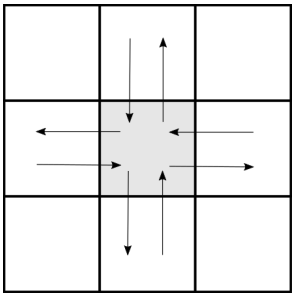
$$\frac{\delta a}{\delta t} = -\mathbf{u} \cdot \nabla a + k_a \nabla^2 - \alpha_a a + S_a, \tag{3.3}$$

kde k_a je difuzní konstanta, α_a je míra rozptylu a S_a odpovídá zdroji, jak bylo uvedeno v předcházející sekci. Narozdíl od Navier-Stokesových rovnic,

obsahuje tato rovnice část difuzní, která je v časovém kroku Δt určena dle:

$$(1 + \Delta t \alpha_a) a(x, t + \Delta t) = a(x, t). \tag{3.4}$$

Pokud v poli hustot existují nenulové hodnoty, s časem se rozptylují po prostoru. Tato difuse probíhá mezi sousedními buňkami mřížky, jak je naznačeno na obrázku 3.4.



Obrázek 3.4: Naznačení difuzního kroku pro centrální buňku ve 2D. Dochází k výměně hodnot hustoty se všemi přímými sousedy.

Přímé rozptylování hodnot hustoty však může při výpočtu vést k nestabilitě simulace, proto je pro zajištění stability nezbytné provádět výpočet zpětným krokem. Cílem výpočtu je nalézt hodnoty hustoty, které při zpětném difuzním kroku dají hodnoty počáteční. K řešení vzniklé soustavy lineárních rovnic je možné použít iterativní postup, např. Gauss-Seidelovu relaxaci. Tato metoda pak odpovídá řešení rovnice 3.4.

Hustoty se dále šíří prostorem mřížky v závislosti na rychlostním poli (advekce). Obdobně jako v případě difuze by bylo možné použít Gauss-Seidelovu relaxaci, prakticky však není tento přístup příliš vhodný, neboť výsledná soustava by závisela na rychlosti a výpočet by byl časově náročnější. Namísto toho je použita metoda jiná, při které je každá buňka mřížky zastoupena pomyslnou částicí ve svém středu, kterou je možno pohybovat polem rychlostí. V každém časovém kroku jsou pro každou buňku určeny částice, které by skončily přesně ve středu buňky. Toto schéma je znázorněno na obrázku 3.5. Množství hustoty, kterou tyto částice nesou, odpovídá lineární interpolaci hodnot z nejbližších sousedních buněk. Advekční krok lze zapsat jako:

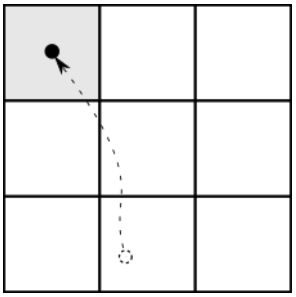
$$a_{advected}(x) = a_{not\_advected}(\mathbf{p}(x, -\Delta t)), \tag{3.5}$$

kde $\mathbf{p}(x, -\Delta t)$ je původní pozice v čase o Δt menším určená dle rychlostního pole. Je zřejmé, že tento výpočet vyžaduje při implementaci druhou mřížkovou strukturu, do které budou uloženy nově spočtené hodnoty hustoty.

3.1.3 Výpočet rychlostního pole

Vývoj rychlostního pole může být popsán rovnicí, která je téměř totožná s první z Navier-Stokesových rovnic:

$$\frac{\delta \mathbf{u}}{\delta t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} + \nu \nabla^2 \mathbf{u} + \mathbf{f}, \tag{3.6}$$



Obrázek 3.5: Naznačení advekce v levé horní buňce ve 2D. Částice, která při pohybu rychlostním polem skončí přesně v jejím středu, se nacházela na pozici znázorněné přerušovaně. Hodnota hustoty v levé horní buňce je určena lineární interpolací hodnot sousedních buněk na původní pozici.

kde člen $-(\mathbf{u} \cdot \nabla)\mathbf{u}$ představuje ve výpočtu advekční krok, člen $\nu\nabla^2\mathbf{u}$ difuzi a $\mathbf{f}$ přidání zdroje rychlosti.

Podobně jako při výpočtu pole hustot, se i v poli rychlostí aplikuje na hodnoty difuse:

$$(\mathbf{I} - \mu\Delta t\nabla^2)\mathbf{u}_{diffused}(x) = \mathbf{u}_{not\_diffused}(x). \tag{3.7}$$

Zde je opět možné použít Gauss-Seidelovu relaxaci. Rychlostní pole by se dále mělo pohybovat podle sebe sama. V této části je možné znovu využít advekční schéma uvedené v předcházející sekci, tentokrát pro rychlost:

$$\mathbf{u}_{advected}(x) = \mathbf{u}_{not\_advected}(\mathbf{p}(x, -\Delta t)). \tag{3.8}$$

Při výpočtu rychlostního pole však dochází navíc k výpočtu projekce. Projekce vynucuje na rychlostním poli požadavek na zachovávání hmotnosti, významné vlastnosti reálných kapalin. Vizuálně umocňuje přítomnost vírů v toku a přibližuje tak simulaci blíže realitě.

Při výpočtu pojekce je využito poznatků Hodgeovy dekompozice, dle které je každé rychlostní pole $\mathbf{w}$ součtem pole zachovávajícího hmotnost $\mathbf{u}$ a skalárního pole q [14]:

$$\mathbf{w} = \mathbf{u} + \nabla q \tag{3.9}$$

Cílem je získat pole $\mathbf{u}$ zachovávající hmotnost, je tedy zaveden projekční operátor P:

$$\mathbf{u} = \mathbf{P}\mathbf{w}. \tag{3.10}$$

Tento operátor odpovídá úpravě původní rovnice 3.9 na tvar

$$\nabla \cdot \mathbf{w} = \nabla^2 q, \tag{3.11}$$

což odpovídá Poissonově rovnici pro skalární pole q. Při výpočtu je možné opět využít Gauss-Seidelovu relaxaci, výsledku je využito při určení projekce $\mathbf{u}$ následovně:

$$\mathbf{u} = \mathbf{P}\mathbf{w} = \mathbf{w} - \nabla q, \tag{3.12}$$

neboli

$$\mathbf{u}_{projected}(x) = \mathbf{u}_{not\_projected}(x) - \nabla q, \tag{3.13}$$

což znamená, že jsme spočítali pole gradientů, které je odečteno od stávajícího rychlostního pole. Pokud je operátor aplikován na obě strany první z Navier-Stokesových rovnic (2.2), dostaneme pro rychlostní pole jednu rovnici ve tvaru:

$$\frac{\delta \mathbf{u}}{\delta t} = \mathbf{P}\left(-(\mathbf{u}\cdot\nabla)\mathbf{u} + \nu\nabla^2\mathbf{u} + \mathbf{f}\right), \tag{3.14}$$

kde je využito faktu, že $\mathbf{P}\mathbf{u} = \mathbf{u}$ a $\mathbf{P}\nabla p = 0$.

3.1.4 Hraniční podmínky

Při aplikaci mřížkové metody je nutné správně nastavit hraniční podmínky. Podmínky obvykle spočívají v obklopení simulovaného prostoru neprostupnou hranicí tak, aby žádná hustota nemohla opustit prostor mřížky. V tomto případě stačí vynutit, aby v hraničních buňkách normálová složka rychlosti byla nulová (normálová vzhledem k hranici; např. pro horizontální hranici je normálovou složkou složka vertikální).

Hraniční podmínky se však nemusí omezovat pouze na okraj mřížky, ale lze je aplikovat v různé podobě v celém prostoru simulační mřížky, jako hranice pevných objektů apod. Např. článek [15] pracuje i s pohyblivými hranicemi.

3.2 SPH

Během rešeršních prací se ukázalo, že mnoho simulací lavin je založeno na metodě Smoothed Particle Hydrodynamics. SPH je systémem částic, ve kterém každá částice nese informaci o kapalině v malé oblasti [18]. Nesenou informací může být např. hustota nebo rychlost. V průběhu simulace pak dochází k úpravě hodnot na základě interakcí mezi částicemi. Úprava hodnot spočívá v sečtení přírůstků ostatních částic, které jsou váženy jádrovou funkcí. Použití této funkce zajišťuje, že přírůstek vzdálenějších částic je nižší, než přírůstek částic bližších [21]. Typický průběh simulace SPH je uveden na obrázku 3.6.

Důležitým parametrem v SPH je parametr h, který určuje nejzazší vzdálenost od částice, ve které se může nacházet částice jiná, aby byl započten její přírůstek.

Jednotlivé skalární (např. hustota) i vektorové (např. tlaková síla, viskozita) hodnoty $\mathbf{A}(\mathbf{p})$ na pozici $\mathbf{p}$ jsou obecně vypočteny jako součet přírůstků hodnot sousedních částic $\mathbf{A_i}$ ze vzorce [5]:

$$\mathbf{A}(\mathbf{p_i}) = \sum_j m_j \frac{\mathbf{A_j}}{\rho_j} W(|\mathbf{p_i} - \mathbf{p_j}|, h), \tag{3.15}$$

kde m_j představuje hmotnost sousední částice, ρ_j hustotu sousední částice a W je jádrová funkce (angl. kernel) určující míru přírustku $\mathbf{A_j}$ vzhledem ke vzdálenosti částice na pozici $\mathbf{p_i}$ od sousední částice na pozici $\mathbf{p_j}$.

Ve většině dále uvedených rovnic je potřeba určit i derivace veličin. V SPH mají derivace vliv pouze na jádrovou funkci [18]. Gradient $\mathbf{A}$ je jednoduše:

Obrázek 3.6: Nejčastější průběh simulačního kroku SPH; existují případy, kdy některý z podkroků bude vynechán nebo upraven. Např. akcelerační mřížka nemusí být použita a výpočet působících sil se může skládat z různých kroků. U důležitých kroků je uvedena rovnice, kterou realizují.

$$\nabla \mathbf{A}(\mathbf{p_i}) = \sum_j m_j \frac{\mathbf{A_j}}{\rho_j} \nabla W(|\mathbf{p_i} - \mathbf{p_j}|, h) \tag{3.16}$$

a laplacián $\mathbf{A}$ pak:

$$\nabla^2 \mathbf{A}(\mathbf{p_i}) = \sum_j m_j \frac{\mathbf{A_j}}{\rho_j} \nabla^2 W(|\mathbf{p_i} - \mathbf{p_j}|, h). \tag{3.17}$$

Zmíněná jádrová funkce W se může lišit v závislosti na použití a požadovaném chování, měla by však být normalizovaná [19]. Zároveň jsou na jádrovou funkci kladeny další podmínky [21], které zajišťují správné chování simulace. Patří mezi ně nezápornost funkce v celém definičním oboru, sudost a také omezení vyplývající z již dříve formulovaného požadavku na nulový přírůstek hodnot od částic, které se nachází ve vzdálenosti větší než h. Musí tedy platit:

$$W(\mathbf{r}, h) = 0, |\mathbf{r}| > h. \tag{3.18}$$

Zde zavedená proměnná $\mathbf{r}$ by při výpočtu odpovídala rozdílu pozic částic, tedy

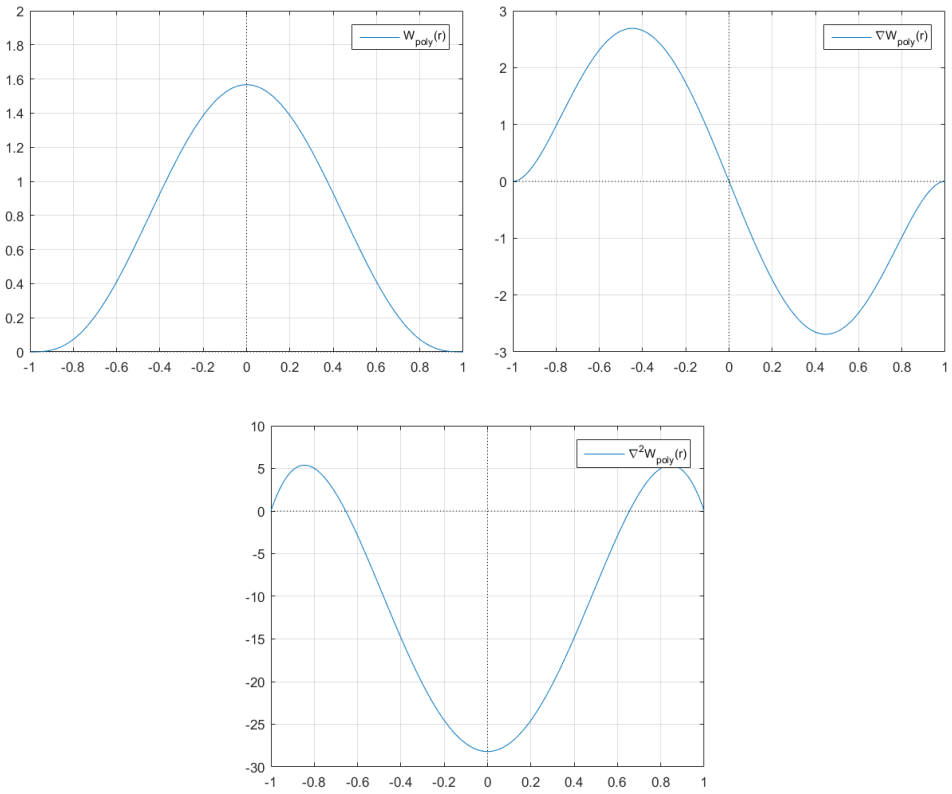
$$\mathbf{r} = \mathbf{p_i} - \mathbf{p_j}. \tag{3.19}$$

V praxi často používané tzv. standardní jádro je výpočetně rychlé, neboť stačí znát vzdálenost $\mathbf{r}$ v druhé mocnině $|\mathbf{r}|^2$. V základním tvaru pak jádrová funkce odpovídá:

$$W_{poly}(\mathbf{r}, h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - |\mathbf{r}|^2)^3, & 0 \leq |\mathbf{r}| \leq h \\ 0, & jinak, \end{cases} \tag{3.20}$$

gradient:

$$\nabla W_{poly}(\mathbf{r}, h) = -\frac{945}{32\pi h^9} (h^2 - |\mathbf{r}|^2)^2 \mathbf{r}, \quad 0 \leq |\mathbf{r}| \leq h, \tag{3.21}$$



Obrázek 3.7: Průmět standardního jádra do jedné dimenze pro $h = 1$. Uveden postupně základní tvar $W_{poly}(\mathbf{r}, h)$, gradient $\nabla W_{poly}(\mathbf{r}, h)$ a laplacián $\nabla^2 W_{poly}(\mathbf{r}, h)$.

a laplacián:

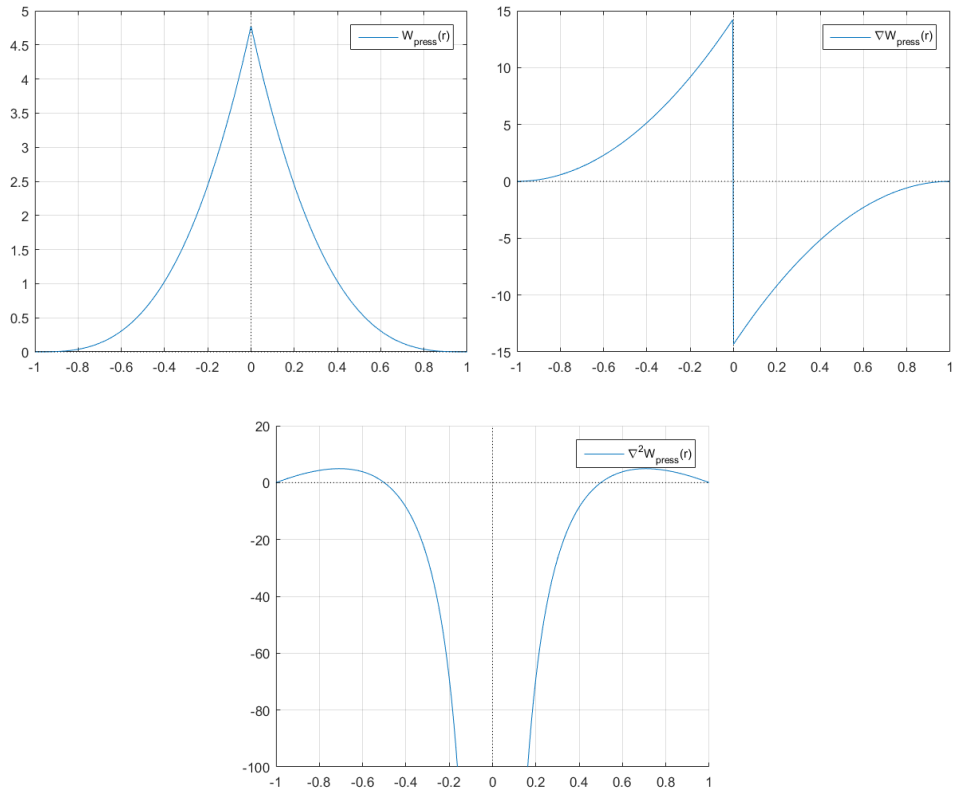
$$\nabla^2 W_{poly}(\mathbf{r}, h) = -\frac{945}{32\pi h^9}(h^2 - |\mathbf{r}|^2)(3h^2 - 7|\mathbf{r}|^2), \quad 0 \leq |\mathbf{r}| \leq h. \tag{3.22}$$

Pro výpočet sil působících v kapalině je nutno znát hustotu kapaliny. K jejímu výpočtu se hodí právě uvedená jádrová funkce a celý výpočet hodnoty hustoty v bodě $\mathbf{p_i}$ pak vypadá následovně:

$$\rho(\mathbf{p_i}) = \sum_j m_j W_{poly}(\mathbf{r}, h). \tag{3.23}$$

Pro výpočet tlakového pole se výše uvedená jádrová funkce $W_{poly}(\mathbf{r}, h)$ nehodí, neboť její derivace je nulová pro hodnoty $\mathbf{r}$ jdoucí k nule. To odporuje požadovanému chování, kdy by na sebe měly dvě velmi blízké částice působit odpudivou silou tím vyšší, čím blíže k sobě se nachází. Z tohoto důvodu je používáno speciální jádro pro tlakové pole $W_{press}(\mathbf{r}, h)$:

$$\nabla W_{press}(\mathbf{r}, h) = \frac{15}{\pi h^6} \begin{cases} (h - |\mathbf{r}|)^3, & 0 \leq |\mathbf{r}| \leq h \\ 0, & jinak, \end{cases} \tag{3.24}$$



Obrázek 3.8: Průmět jádra pro tlakové pole do jedné dimenze pro $h = 1$. Uveden postupně základní tvar $W_{press}(\mathbf{r}, h)$, gradient $\nabla W_{press}(\mathbf{r}, h)$ a laplacián $\nabla^2 W_{press}(\mathbf{r}, h)$.

s gradientem (uplatňuje se při výpočtu tlakové síly):

$$\begin{gathered} \nabla W_{press}(\mathbf{r}, h) = -\frac{45}{64\pi h^6}\frac{\mathbf{r}}{|\mathbf{r}|}(h - |\mathbf{r}|)^2, \\ \lim_{r\to 0^-} \nabla W_{press}(r, h) = \frac{45}{64\pi h^6}, \quad \lim_{r\to 0^+} \nabla W_{press}(r, h) = -\frac{45}{64\pi h^6} \end{gathered} \tag{3.25}$$

a laplaciánem:

$$\begin{gathered} \nabla^2 W_{press}(\mathbf{r}, h) = -\frac{90}{\pi h^6}\frac{1}{|\mathbf{r}|}(h - |\mathbf{r}|)(h - 2|\mathbf{r}|), \\ \lim_{r\to 0} \nabla^2 W_{press}(r, h) = -\infty. \end{gathered} \tag{3.26}$$

Celý výpočet tlakové síly působící v bodě $\mathbf{p_i}$ je pak možno zapsat následující rovnicí:

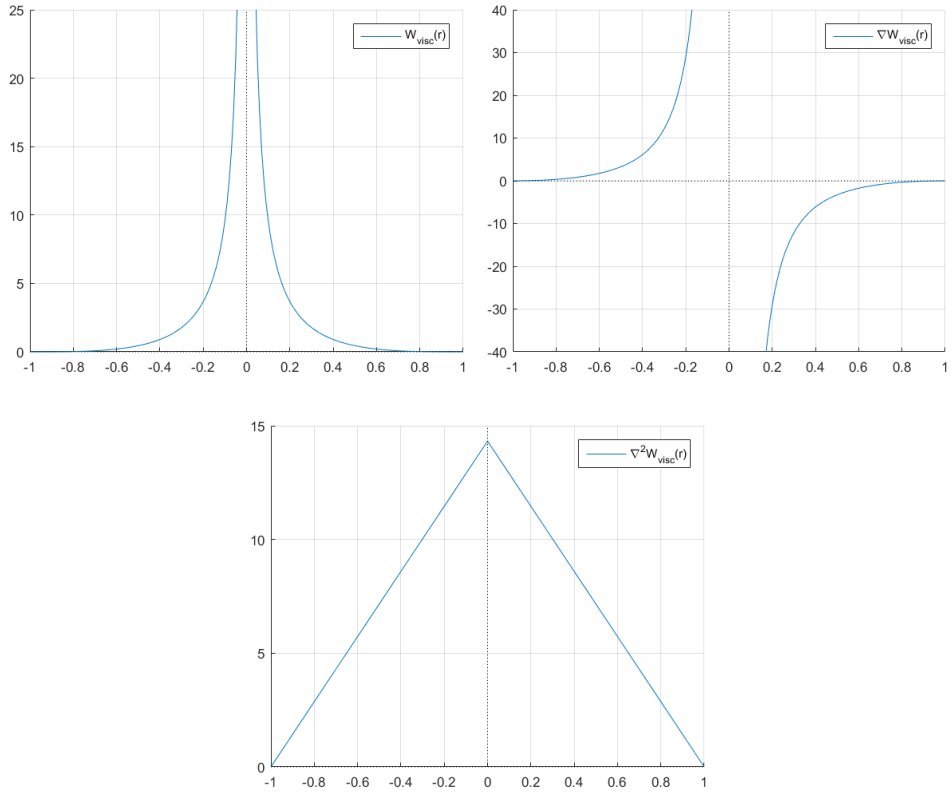
$$\mathbf{f_i^{press}} = \sum_j m_j \frac{P_i + P_j}{2\rho_j} \nabla W_{press}(\mathbf{r}, h), \tag{3.27}$$

kde P_i a P_j odpovídají tlaku v bodě $\mathbf{p_i}$, respektive $\mathbf{p_j}$. Součet a podíl jsou použity, aby rovnice produkovala symetrické síly (bez použití symetrizace

by výsledné síly v bodech $\mathbf{p_i}$, respektive $\mathbf{p_j}$ byly rozdílné, pokud by v těchto bodech byly různé hodnoty tlaku). Tlak je obvykle počítán z rovnice odvozené ze stavové rovnice ideálního plynu. Odvozená rovnice má tvar:

$$P = k(\rho - \rho_0), \tag{3.28}$$

kde k je konstanta tuhosti a ρ_0 odpovídá klidové hustotě modelované kapaliny. Je však nutné uvést, že jiní autoři používají k výpočtu tlaku i odlišné přístupy, např. Poissonovu nebo Taitovu rovnici [20].



Obrázek 3.9: Průmět jádra pro viskozitu do jedné dimenze pro $h = 1$. Uveden postupně základní tvar $W_{visc}(\mathbf{r}, h)$, gradient $\nabla W_{visc}(\mathbf{r}, h)$ a laplacián $\nabla^2 W_{visc}(\mathbf{r}, h)$.

Simulace kapalin pomocí metody SPH obvykle zahrnuje i simulaci odporu kapaliny k tečení, tzv. viskozity [21]. Obdobně jako v případě tlakových sil, ani v případě viskozity není použití standardního jádra vhodné a nevyhovuje ani jádro pro tlakové pole. Pro výpočet viskozity je potřeba laplacián jádra, přičemž obě zmíněné jádrové funkce mají pro část (nebo celý) definiční obor záporný laplacián. Proto je zavedena speciální jádrová funkce pro viskozitu:

$$W_{visc}(\mathbf{r}, h) = \frac{15}{2\pi h^3} \begin{cases} -\frac{|\mathbf{r}|^3}{2h^3} + \frac{|\mathbf{r}|^2}{h^2} + \frac{h}{2|\mathbf{r}|} - 1, & 0 \leq |\mathbf{r}| \leq h \\ 0, & jinak, \end{cases} \tag{3.29}$$

$$\lim_{r \to 0} W_{visc}(r, h) = \infty.$$

s gradientem:

$$\nabla W_{visc}(\mathbf{r},h) = \frac{15}{2\pi h^3}\mathbf{r}\left(-\frac{3|\mathbf{r}|}{2h^3}+\frac{2}{h^2}-\frac{h}{2|\mathbf{r}|^3}\right), \\ \lim_{r\to 0^-}\nabla W_{visc}(r,h) = +\infty, \quad \lim_{r\to 0^+}\nabla W_{visc}(r,h) = -\infty. \tag{3.30}$$

a s laplaciánem ve tvaru (uplatňuje se ve výpočtu viskózních sil):

$$\nabla^2 W_{visc}(\mathbf{r},h) = \frac{45}{\pi h^6}(h-|\mathbf{r}|). \tag{3.31}$$

Jako u tlakové síly je i v případě viskozity nutné zajistit symetrický výpočet. Dle [18] závisí viskozita pouze na relativní rychlosti částic, dosazením do rovnice 3.17 tedy dostaneme výsledný vztah:

$$\mathbf{f_i^{visc}} = \mu\sum_j m_j\frac{\mathbf{v_i}+\mathbf{v_j}}{\rho_j}\nabla^2 W_{visc}(\mathbf{r},h). \tag{3.32}$$

Někteří autoři výpočet SPH rozšiřují o další síly, jmenovitě např. povrchové napětí [21]. Povrchové napětí má významný vliv na simulaci v případě kapalin, zkoumané zdroje zabývající se simulací lavin však jeho výpočet přímo nezmiňují. Jiní autoři [25, 26] pak uvedený výpočet viskozity 3.32 nahrazují výpočtem tenzoru namáhání (stress) $\mathbf{S}$ a výpočtem příslušné síly $\mathbf{f_i^{stress}}$ dle vztahu [26]:

$$\mathbf{f_i^{stress}} = \sum_j m_j\frac{\mathbf{S}_i+\mathbf{S}_j}{\rho_i\rho_j}\cdot\nabla W_{stress}(\mathbf{r},h), \tag{3.33}$$

$$\mathbf{S} = \eta(D)\mathbf{D}, \tag{3.34}$$

$$D = \sqrt{\frac{1}{2}\cdot trace(\mathbf{D}^2)}, \tag{3.35}$$

$$\eta(D) = (1-exp[-(J+1)D])\left(D^{n-1}+\frac{1}{D}\right), \tag{3.36}$$

kde J je parametr určující míru viskozity a $\mathbf{D}$ je tenzor rychlosti deformace. Tenzor $\mathbf{D}$ pro částici i je určen jako:

$$\mathbf{D}_i = \nabla\mathbf{v}_i + (\nabla\mathbf{v}_i)^T, \tag{3.37}$$

přičemž rychlostní tenzor $\nabla\mathbf{v}_i$ je určen ze vztahu:

$$\nabla\mathbf{v}_i = \sum_j\frac{m_j}{\rho_j}(\mathbf{v}_j-\mathbf{v}_i)\otimes W_{stress}(\mathbf{r},h), \tag{3.38}$$

kde operace $\otimes$ představuje tenzorový součin, který je dle [27] určen jako:

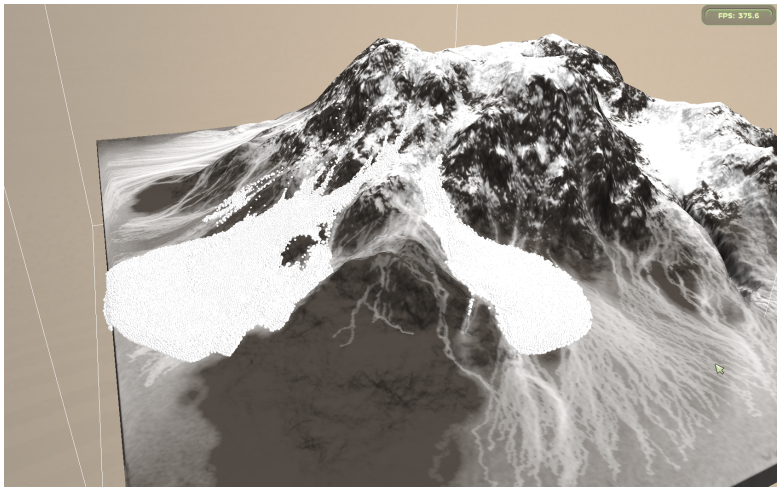
$$\mathbf{T} = \mathbf{u}\otimes\mathbf{v} \tag{3.39}$$

$$\mathbf{T}_{i,j} = \mathbf{u}_i\mathbf{v}_j, \quad i,j\in\{1,2,3\}. \tag{3.40}$$

Uvedený kernel $W_{stress}(\mathbf{r},h)$ není v případě práce [25] specifikován, neboť je zkoušeno více různých kernelů. V případě zdroje [26] se jedná o speciální kernel,

který je použit ve všech výpočtech (nahrazuje $W_{poly}(\mathbf{r}, h)$ a $W_{press}(\mathbf{r}, h)$, a vynucuje tak dodatečné použití korekce rychlosti):

$$W_{stress}(\mathbf{r}, h) = \frac{315}{208\pi h^3} \cdot w\left(\frac{|\mathbf{r}|}{h}\right),$$
$$w(q) = \begin{cases} \frac{2}{3} - \frac{9}{8}q^2 + \frac{19}{24}q^3 - \frac{5}{32}q^4, & 0 \leq q \leq 2 \\ 0, & jinak. \end{cases} \tag{3.41}$$



Obrázek 3.10: Ukázka simulace laviny pomocí metody SPH. Převzato z [25].

Příspěvky všech dílčích sil působích na částici jsou následně při výpočtu sečteny do síly výsledné. Tento součet probíhá ve všech případech, ať je použit tenzorový výpočet 3.33 jednodušší výpočet viskozity dle 3.32 nebo výpočet jiných sil. Z výsledné síly je spočteno zrychlení částice, ke kterému je přičteno zrychlení gravitační. Integrací zrychlení podle časového kroku jsou pak určeny nové hodnoty rychlosti a pozice částice.

Kolize s terénem jsou řešeny posunem částice do bodu kontaktu (pokud pronikla pod terén) a ozrcadlením vektoru rychlosti částice okolo normály terénu v bodě kontaktu.

Většina autorů při implementaci SPH využívá pomocné mřížky (pozn. jedná se o akcelerační mřížku a nikoliv o mřížku popsanou v sekci 3.1). Vzhledem k faktu, že je přírůstek hodnot pro jednotlivé částice počítán pouze z malého množství nejbližších sousedních částic, může být použitím akcelerační struktury výpočet urychlen. Používaná uniformní mřížka dělí prostor do stejně velkých buněk, přičemž minimální délka hrany buňky musí odpovídat parametru h (aby bylo zajištěno, že bude započítán přírůstek všech částic ve vzdálenosti menší, než je hodnota tohoto parametru). V každém kroku simulace je test vzdálenosti a případný následný výpočet prováděn pouze nad částicemi obsaženými v nejbližších 27 buňkách k upravované částici (včetně buňky, ve které je tato částice obsažena). Nutno poznamenat, že mnoho autorů [22, 23] namísto uniformní mřížky využívá prostorového hashování (původně představeno v [24]), které umožňuje hledání sousedních částic dále zrychlit.

Práce [4] pracuje s rozšířeným modelem SPH pro simulaci laviny tvořené prachovým sněhem. Autor vychází z předpokladu, že vývoj takové laviny se řídí přítomností vzduchu a jeho strháváním do pohybujícího se sněhu. Takový tok, který se řídí přítomností dvou odlišných kapalin, je v mechanice kapalin nazýván tokem dvoufázovým. Použití matematického modelu pro simulaci dvoufázového toku by vedlo na řešení dvou rychlostních polí a bylo by tak výpočetně náročné. Je proto navrženo zjednodušení na jedno rychlostní pole vycházející z předpokladu, že jsou obě fáze nuceny řídit se stejnou rychlostí.

3.3 Kombinované metody

Přestože je možné sníh a laviny simulovat s různými úpravami jen za samostatného použití metod dosud uvedených, častějším přístupem je snaha využít výhod jak Lagrangeovského přístupu (částice, SPH), tak přístupu Newtonovského (mřížka). Kombinací obou metod je možné dosáhnout věrnější reprodukce reálného chování sněhu, vyvstávají však také nové problémy, zejména při vzájemné interakci obou metod, kdy mřížka je omezena na fixní diskretizaci prostoru, zatímco částice nikoliv. Následuje představení několika metod, které kombinovaného systému využívají k dosažení realitěbližších výledků.

3.3.1 Metoda interakcí mezi vrstvami

V práci [5] je navržena metoda pro simulaci lavin kombinovaného typu. Základní těleso laviny o vyšší hustotě odpovídající mokrému sněhu je simulováno pomocí SPH, vrstva zvířeného prachového sněhu pak mřížkovou metodou. Toto rozdělení vychází z požadovaného chování obou vrstev: zatímco základní vrstva se chováním blíží kapalině a je podrobena velkému množství kolizí s terénem, vrstva prachového sněhu připomíná plyn a pro výpočet jeho detailů se více hodí přístup s mřížkou.

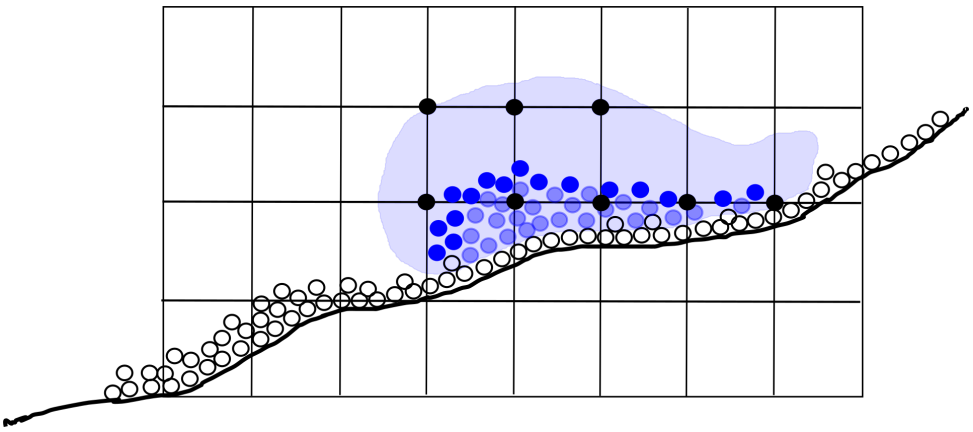
Aby mohla simulace laviny využít kombinace SPH i mřížkové metody, je zde zavedena tzv. oblast překrytí, ve které existuje jak sníh představovaný částicemi, tak prachový sníh generovaný mřížkou. Oblast překrytí je definována jako oblast, kde je objemová hustota částic[1] nižší, než ve zbytku základní vrstvy laviny (zjednodušeně lze říci, že se oblast překrytí nachází při povrchu tekoucí laviny). Objemová hustota na pozici $\mathbf{p}_i$ je určena z následující rovnice:

$$n(\mathbf{p}_i) = \sum_j W_n(|\mathbf{r}_j - \mathbf{p}_i|, h) \tag{3.42}$$

$$W_n(\mathbf{r}, h) = \begin{cases} \frac{h}{\mathbf{r}} - 1 & 0 \leq |\mathbf{r}| < h, \\ 0 & \mathbf{r} > h. \end{cases} \tag{3.43}$$

V této vrstvě pak probíhá generování prachového sněhu (kdy je část hmotnosti částic přenesena do mřížky), adheze prachového sněhu (kdy je množství sněhu

[1] number density



Obrázek 3.11: Princip kombinované metody. Bílé částice tvoří akumulovanou vrstvu při povrchu terénu a po stržení se mohou stát částicemi vrstvy tekoucí (modré částice). Do interakce s vrstvou prachového sněhu (černá mřížka, černé body obsahují nenulovou hustotu) vstupují pouze určité částice tekoucí vrstvy (tmavě modrá).

přilnutého k částici určeno z objemu, kterým se částice v časovém kroku pohybuje) a výpočet interakčních sil.

Při generování prachového sněhu je část masy interagující částice přenesena na mřížku. V popsané metodě zůstává při interakci velikost částice neměnná a mění se pouze její hmotnost. Množství sněhu G_j přenesené na buňku mřížky j je modelováno dle vztahu:

$$G_j = \sum_i C_g |\mathbf{u}_j - \mathbf{v}_i|^{1.7} S_i^{sur} \Delta t, \tag{3.44}$$

kde C_g je konstanta určující míru generování sněhu, $\mathbf{u}_j$ je rychlost v buňce j mřížky, v_i je rychlost částice i sousedící s buňkou j a S_i^{sur} odpovídá povrchu částice i.

Naopak adheze prachového sněhu je modelována jako množství, které částice SPH v daném časovém kroku smete:

$$A_i = \sum_j C_a d_j S_i^{sec} |\mathbf{u}_j - \mathbf{v}_i| \Delta t, \tag{3.45}$$

kde C_a je konstanta určující míru adheze, d_j je hustota v buňce j mřížky sousedící s částicí i a S_i^{sec} odpovídá ploše kruhového průřezu částice i.

Interakční síly působící na částici i z buňky mřížky j jsou vypočteny dle následujících vztahů:

$$\mathbf{f}_{i,j}^{drag} = -k_{drag} \frac{m_i}{r_i} |\mathbf{v}_i - \mathbf{u}_i| (\mathbf{v}_i - \mathbf{u}_i), \tag{3.46}$$

$$\mathbf{f}_{i,j}^{lift} = -k_{lift} m_i (\mathbf{v_i}_i - \mathbf{u}_i) \times \omega_i, \tag{3.47}$$

kde k_{drag} a k_{lift} jsou konstanty (zvoleny 0.81 a 0.2), m_i je hmotnost částice i, r_i její poloměr a $\mathbf{v}_i$ její rychlost. $\mathbf{u}_i$ je rychlost na pozici částice i interpolovaná ze sousedních buněk mřížky. Obdobně ω_i je vířivost na pozici i interpolovaná

z vířivostí sousedních buňek, přičemž vířivost v buňce j je spočtena jako $\omega_j^{\mathbf{g}} = \nabla \times \mathbf{u}_j^{\mathbf{g}}$, kde $\mathbf{u}_j^{\mathbf{g}}$ je rychlost v buňce j mřížky. Síly působící z částic na bod mřížky jsou spočteny jako reakční síly $\mathbf{f}_{i,j}^{drag}$ a $\mathbf{f}_{i,j}^{lift}$.

Celková vnější síla působící na buňky mřížky je pak určena dle vztahu:

$$\mathbf{f} = \rho \boldsymbol{g} + \mathbf{f}^{buoyancy} + \mathbf{f}^{lift} + \mathbf{f}^{drag} + \mathbf{f}^{conf}, \tag{3.48}$$

kde $\mathbf{g}$ představuje gravitační vektor, $\mathbf{f}^{buoyancy}$ je vztaková síla určená ze vztahu:

$$\mathbf{f}^{buoyancy} = -(1 - r_s)\rho \boldsymbol{g}, \tag{3.49}$$

$$r_s = \frac{\rho_{smoke}}{\rho_{air} + \rho_{smoke}}, \tag{3.50}$$

kde r_s určuje poměr mezi hustotou vzduchu a hustotou simulované látky. $\mathbf{f}^{conf}$ je síla generující rotace a víry pomocí metody vorticity confinement. Cílem této metody je navrátit do rychlostního pole detaily, které se ztrácí při nefyzikálním numerickém řešení simulace [31].

$$\mathbf{f}^{conf} = \epsilon(\boldsymbol{N} \times \boldsymbol{\omega}_i) \tag{3.51}$$

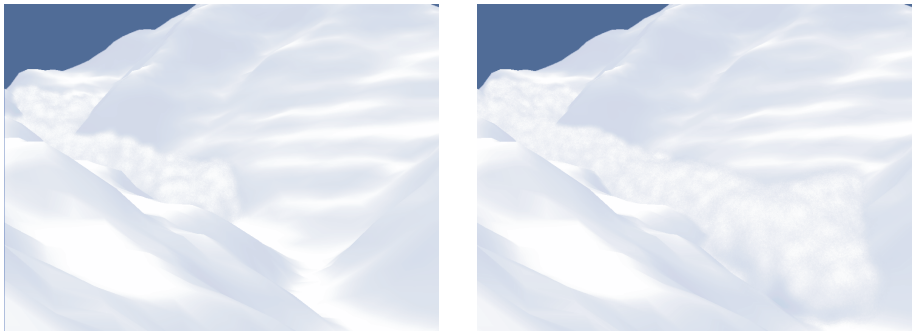
Množtsví detailů, které bude navráceno do rychlostního pole lze řídit konstantou $\epsilon > 0$. Normalizované vektory $\mathbf{N}$, které směřují z bodů s nižšími koncentracemi vířivosti do bodů s koncentracemi vyššími, jsou určeny jako

$$\boldsymbol{N} = \frac{\boldsymbol{\eta}}{|\boldsymbol{\eta}|}, \tag{3.52}$$

$$\boldsymbol{\eta} = \nabla|\boldsymbol{\omega}|, \tag{3.53}$$

a $\boldsymbol{\omega}$ představuje vířivost určenou z rychlosti $\boldsymbol{u}$ následovně:

$$\boldsymbol{\omega} = \nabla \times \boldsymbol{u}. \tag{3.54}$$



Obrázek 3.12: Výstup kombinované metody ve 3D, převzato z [5].

Práce [5] představující zde uvedenou kombinovanou metodu navíc pracuje s rozšířením základní části tvořené systémem SPH. Toto rozšíření spočívá v rozdělení částic na dva různé typy. První typ částic odpovídá klasickým částicím popsaným v sekci 3.2 a tvoří základní pohybující se těleso laviny. Druhý typ částic reprezentující sníh na povrchu terénu se nesmí pohybovat,

dokud nejsou splněny podmínky odtržení částice. Jakmile jednou k odtržení dojde, částice se chová jako částice prvního typu. Tímto způsobem je možno simulovat strhávání sněhu pohybující se lavinou, její adekvátní nárůst objemu i zanechání stopy ve sněhu pokrývajícím terén. Tato vrstva tvořená sněhem, který se nepohybuje dokud nedojde k jeho stržení, je nazvána jako vrstva akumulovaného sněhu.

Podmínkou pro odtržení částice je dosažení dostatečné deformační rychlosti (strain rate) $\boldsymbol{\gamma}_i$:

$$\boldsymbol{\gamma}_i = \frac{1}{\boldsymbol{\tau}} W_i \sin\theta + \boldsymbol{\gamma}_f, \tag{3.55}$$

$$\boldsymbol{\tau} = 8.5 \times 10^6 exp(0.021\rho_a), \tag{3.56}$$

kde $\boldsymbol{\tau}$ je koeficientem viskozity sněhu v akumulované vrstvě, W_i představuje hmotnost sněhu nacházejího se nad částicí i (je uvažován sníh z akumulované a tekoucí vrstvy; hmotnost prachového sněhu není uvažována, protože je zanedbatelná), θ je úhel svahu, $\boldsymbol{\gamma}_f$ je rychlost deformace v důsledku tření a ρ_a je počáteční vzálenost mezi částicemi tekoucí vrstvy. Při výpočtu W_i se autoři odkazují na článek, který se nepodařilo dohledat.

3.3.2 FLIP

Článek [29] představuje další metodu založenou na kombinaci částic a mřížky. Přestože se práce zabývá simulací sněhu v měřítku spíše menším, měly by být zde představené myšlenky aplikovatelné i při simulaci lavin ve velkém měřítku. Přínos spočívá zejména v možnosti simulovat stlačitelnost sněhu a dále tření a kohezi.

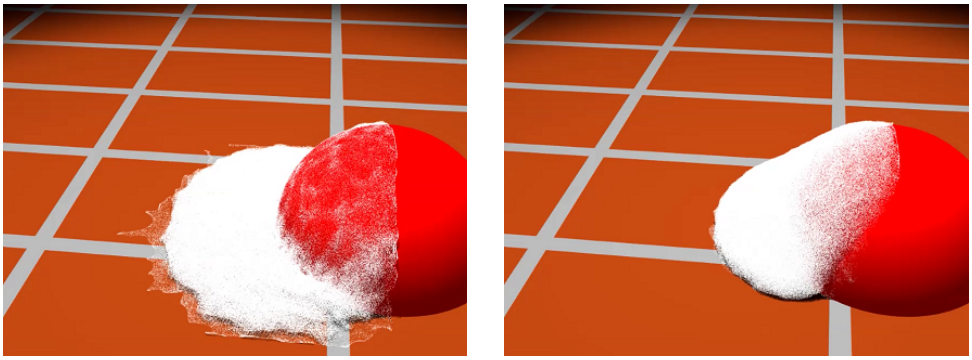
Zde představený způsob nepracuje, narozdíl od dříve uvedené kombinované metody, s částicovým systémem modelovaným pomocí SPH, ale je založen na metodě FLIP. Autoři uvádí, že je FLIP použit proto, že je narozdíl od SPH nebo DEM, jeho výstup nezávislý na distibuci částic. Informace o simulované kapalině jsou v metodě FLIP neseny částicemi, ale během výpočtu jsou dočasně přeneseny na mřížku. Proměnné částic jsou následně upraveny dle změn vypočítaných na mřížce [30].

Sníh je stlačitelný, protože ve svém objemu obsahuje množství vzduchu, které je při stlačování vytlačeno ven, čímž dochází ke snížení objemu sněhu [5]. Tento jev je simulován pomocí speciálních pórovitých částic, které nesou informaci o množství neseného vzduchu. Při výpočtu je charakteristika částice zjednodušena na proměnnou $d, 0 \leq d \leq 1$ označovanou trvanlivost, která udává, nakolik zůstává původní sněhová struktura (obsahující i vzduch) neporušena.

Trvanlivost se dále uplatňuje i při výpočtu kohezního kritéria na mřížce:

$$c = c_0 + k_c(d_0 - d), \tag{3.57}$$

kde c_0 je počáteční kohezní kritérium a k_c je koeficient změny koheze. V závislosti na kohezním kritériu a tlaku jsou buňky mřížky rozděleny na plastické a tuhé. V buňkách označených jako plastické kapalina teče a část energie se ztrácí v důsledku tření. Jelikož tření vzrůstá s tlakem, je použita rovnice



Obrázek 3.13: Vliv změny koeficientu k_f na výstup simulace sněhu prezentované v [29]. Převzato z téhož zdroje.

$$\mu = \mu_0 + k_f P \tag{3.58}$$

k výpočtu koeficientu tření μ. Použité parametry představují počáteční koeficient tření (μ_0) a koeficient změny tření (k_f).

3.3.3 MPM

Kombinovanou metodou pro simulaci sněhu se zabývá i práce [32]. V tomto případě je použita metoda Material point method (MPM), rozšíření dříve uvedené metody FLIP. Oproti FLIPu umožňuje MPM řízení dynamiky pomocí obecnějších konstitutivních modelů[2].

Práce se zaměřuje na speciální vlastnosti sněhu, který obvykle vykazuje vlastnosti jak kapalin, tak pevných látek. Navržený konstitutivní model odpovídá:

$$\Psi(\boldsymbol{F}_E, \boldsymbol{F}_P) = \mu(\boldsymbol{F}_P)||\boldsymbol{F}_E - \boldsymbol{R}_E||_F^2 + \frac{\lambda(\boldsymbol{F}_P)}{2}(J_E - 1)^2, \tag{3.59}$$

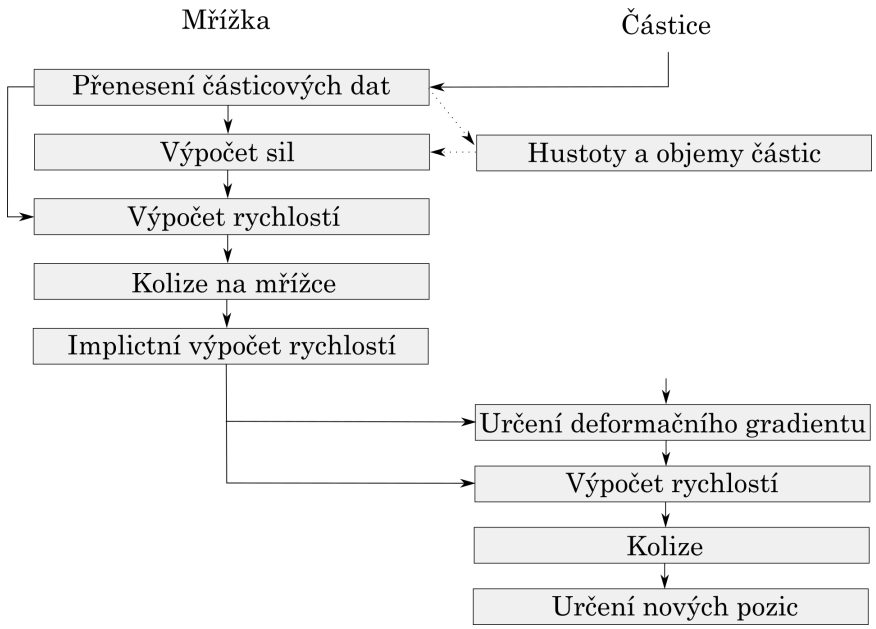
kde deformační gradient $\boldsymbol{F}$ je rozdělen na pružnou část $\boldsymbol{F}_E$ a plastickou část $\boldsymbol{F}_E$ tak, že $\boldsymbol{F} = \boldsymbol{F}_E\boldsymbol{F}_P$. Pružná část je dána fixní hustotou energie a Lamého parametry μ, λ jsou funkcemi plastických deformačních gradientů dle

$$\mu(\boldsymbol{F}_P) = \mu_0 e^{\xi(1-J_P)}, \qquad \lambda(\boldsymbol{F}_P) = \lambda_0 e^{\xi(1-J_P)}, \tag{3.60}$$

kde $J_E = \det(\boldsymbol{F}_E)$, $J_P = \det(\boldsymbol{F}_P)$, $\boldsymbol{F}_E = \boldsymbol{R}_E\boldsymbol{S}_E$, λ_0, μ_0 jsou počátečními Lamého koeficienty a ξ je bezrozměrný parametr plastického tuhnutí. Část pružné a plastické deformace je navíc definována pomocí singulárních hodnot deformačního gradientu. Přidáním kritického tlaku θ_c a tahu θ_s je možné kontrolovat vliv plastických deformací, které se projeví až po překročení určené meze. Změnami uvedených parametrů θ_c, ξ je pak možné simulovat různé druhy sněhu (suchý / mokrý, křehký / plastický, hrudkovitý / prachový).

Celý průběh simulačního kroku pak spočívá v několika dílčích krocích, které kombinují jak Lagrangeovskou část (částice), tak Newtonovskou část (mřížka).

[2] constitutive models



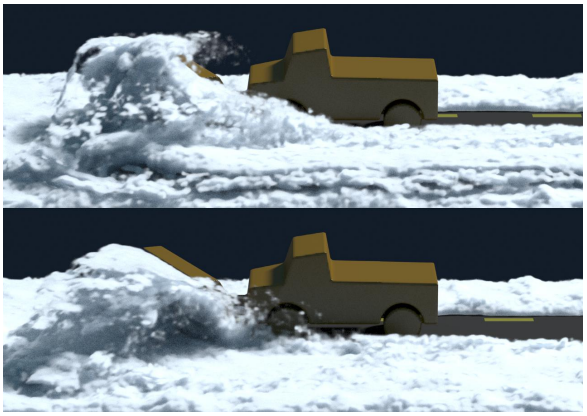
Obrázek 3.14: Schéma průběhu metody uvedené v [32], založené na MPM. Výpočet rychlostí na mřížce se skládá z několika částí, nejprve je z předcházející rychlosti a síly spočtena dočasná rychlost, následně jsou vyřešeny kolize na mřížce a nakonec je vyřešena lineární soustava s dočasnou rychlostí na pravé straně, jejímž řešením je nová hodnota rychlosti.

Jednotlivé kroky jsou graficky znázorněny na schématu 3.14. Nejprve jsou částicová data přenesena na mřížku. Hmotnost a rychlost jsou přeneseny pomocí váhových funkcí (v případě rychlosti se narozdíl od metody FLIP využije normalizovaných váhových funkcí, aby zůstala zachována hybnost). Dále (ale pouze v prvním kroku simulace) jsou určeny hustoty a objemy částic. Následuje výpočet sil na mřížce a patřičná úprava mřížkových rychlostí a reakce na kolize mezi pevnými objekty a mřížkou. V další fázi je upraven deformační gradient pro každou částici, jsou upraveny rychlosti částic, určeny reakce na kolize mezi pevnými objekty a částicemi a nakonec jsou spočteny nové pozice částic.

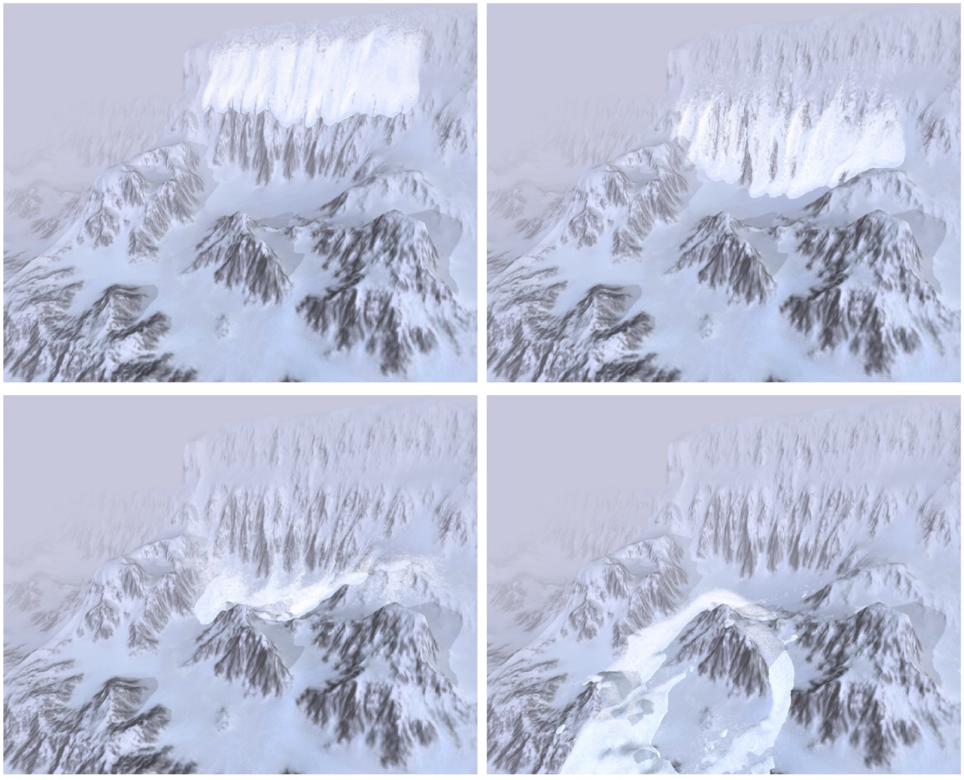
Jak autoři sami zmiňují, metoda v prezentované podobě nezohledňuje interakce se vzduchem, které jsou důležité při simulaci prachového sněhu a při simulaci lavin.

3.4 Molekulární dynamika

V práci [16] je představena metoda simulace lavin založená na molekulární dynamice. Každá částice v simulaci představuje kompaktní kus sněhu o určité hmotnosti, metoda je tedy vhodná spíše pro simulaci lavin tekoucích než lavin vzdušných. Kolize částic jsou řešeny výpočtem normálových a tečných sil. V metodě molekulární dynamiky má zvolený tvar částic vliv na výsledné chování simulace. Např. kulové částice bez dodatečného koeficientu tření,



Obrázek 3.15: Ukázka výstupu metody MPM na simulaci sněžného pluhu. Převzato z [32].



Obrázek 3.16: Ukázka výstupu simulace laviny založené na metodě molekulární dynamiky. Převzato z [16].

který by je držel u země, se na zemi nedokáží stabilizovat a simulace by mohla působit nepřirozeně. Z tohoto důvodu se obvykle používají aproximace složitějších tvarů pomocí koulí [17]. Detekci kolizí mezi jednotlivými částicemi je i v případě nekulových částic možno řešit kulovými obálkami použitých tvarů. Měření vzdálenosti mezi středy obalových kružnic je dostatečné ke zjištění bodů dotyku částic a výrazně usnadňuje výpočet. Výsledné normálové síly působící na částici jsou aplikovány dle normály v bodě dotyku.

Práce [16] dále pracuje s rozšířením uvedené metody v podobě Discrete Element Method (DEM), která do výpočtu přidává pružinové a tlumicí síly. Výsledná kolizní síla působící na částici v tomto rozšířeném modelu je vypočtena dle rovnice 3.61:

$$\mathbf{F_{DEM}} = \mathbf{f_{normal}} + \mathbf{f_{damping}} + \mathbf{f_{shear}} + \mathbf{f_{att}} \tag{3.61}$$

Normálová síla $\mathbf{f_{normal}}$ v uvedené rovnici odpovídá síle působící ve směru normály v bodě kolize dvou částic a její velikost je odvozena od vzdálenosti odpovídající jejich překryvu. Tato síla je zároveň přenásobena pružinovým koeficientem, který do simulace zavádí možnost ovlivňovat stlačitelnost simulované látky.

Tlumicí síla $\mathbf{f_{damping}}$ odpovídá rozdílu rychlostí kolidujících částic přenásobenému tlumicím koeficientem. Velikost koeficientu umožňuje ovlivnit elasticitu srážek v simulaci. Smyková síla $\mathbf{f_{shear}}$ představuje odpor k pohybu v tečném směru.

Uvedená síla $\mathbf{f_{att}}$ odpovídá přitažlivosti mezi částicemi. Použitý koeficient přitažlivosti umožňuje shlukování částic či jejich rozdělování v závislosti na dalším působení externích nebo kolizních sil.

3.5 Zhodnocení metod pro simulaci lavin

Všechny uvedené simulační metody kladou nemalé nároky na výpočetní výkon. V případě Lagrangeovského systému jakým je SPH se úzkým hrdlem stává hledání sousedních částic při výpočtu interakčních sil (zde je možné výpočet částečně urychlit efektivnějším algoritmem, např. akcelerační mřížkou). Výhodou je, že výpočet probíhá pouze v prostoru, který simulovaná látka přímo zaujímá, navíc je výpočetní složitost nezávislá na šíření látky po prostoru, ve kterém simulace probíhá. Tato vlastnost se jeví jako stěžejní pro využití při simulaci lavin, dráha jejichž pohybu může dosahovat enormních rozměrů (v reálném světě v řádu metrů až kilometrů). Pohyb laviny se navíc řídí terénem a oblasti, které budou lavinou zasaženy, nemohou být dopředu přesně známy (naopak, jejich zjištění je jedním z cílů simulace). Lagrangeovské metody jsou z této podstaty pro simulaci vhodné, při zachování počtu částic bude simulace probíhat v řádově stejném čase nezávisle na rozdělení v prostoru.

Oproti tomu Newtonovské metody probíhají v předem definovaném prostoru mřížky a výrazným problémem se tak stává i paměťová náročnost. Pro jednoduchost si představme krychlovou mřížku rozdělenou na $100 \times 100 \times 100$ buněk. Celkový počet buněk v takové mřížce odpovídá 10^6, přičemž se jedná stále o relativně malou mřížku nebo mřížku s relativně nízkým rozlišením (závisí na zvoleném měřítku). Pokud se má lavina pohybovat takovým prostorem, výrazná část mřížky pravděpodobně nebude využita nebo se zde bude nacházet pouze zanedbatelné (a ve vizualizaci neviditelné) množství sněhu. Přesto musí v tomto prázdném prostoru probíhat celý simulační výpočet. Navíc je prakticky nemožné dopředu odhadnout vhodný rozměr mřížky, která by pokryla celý prostor, jímž se bude lavina šířit po celou dobu svého

	Výhody	Nevýhody	Vhodné pro	Odkaz
SPH	Prostorová nezávislost, paměťová náročnost, reakce na kolize	Časová náročnost, nutnost malého časového kroku (stabilita)	Tekoucí laviny	[25]
Molekulární dynamika			Laviny tvořené ledem nebo suchým sněhem	[16]
Mřížka	Časová náročnost, značná nezávislost na časovém kroku	Simulace v přesně definovaném prostoru, paměťová náročnost	Laviny tvořené prachovým sněhem	[14]
Kombinace SPH a mřížky	Komplexní metoda navržená pro laviny	Současné použití dvou přístupů, kombinace jejich nevýhod	Laviny kombinovaného typu	[5]
FLIP, MPM	Přesnější zachycení fyzikálních vlastností sněhu		Detailní sníh o různých vlastnostech	[30, 32]

Tabulka 3.1: Srovnání uvedených metod, jejich výhody, nevýhody a typické použití.

pohybu. Tento problém není v dostupných zdrojích obvykle komentován a simulace probíhá v dopředu pečlivě zvolených scénách. Umožnění simulace laviny v libovolném terénu se tak jeví jako jedna z hlavních výzev této práce.

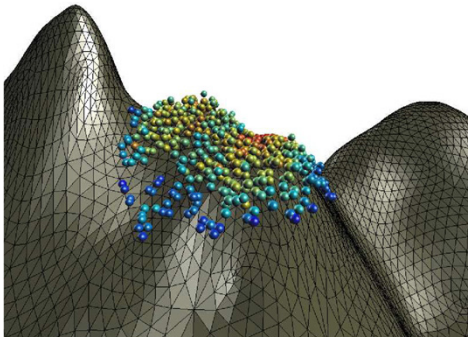
Celkový přehled jednotlivých metod, jejich srovnání a typické použití v kontextu simulace lavin a sněhu je uveden v tabulce 3.1.

3.6 Způsoby vizualizace simulace

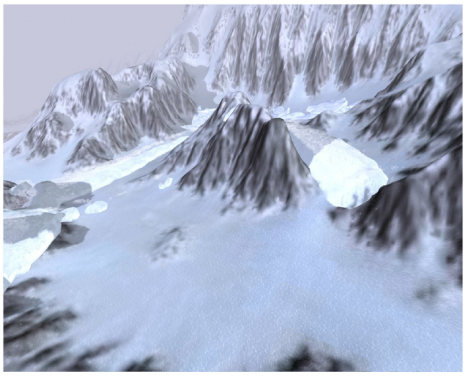
Přestože rozbor vizualizačních technik používaných pro prezentaci výsledků uvedených simulací není primárním cílem této práce, je vhodné uvést alespoň jejich základní přehled.

Nejjednodušším způsobem, jak zobrazit kapalinu Lagrangeovské simulace je vykreslení zástupného objektu na pozici každé částice, obvykle koule nebo otexturované plošky, viz 3.17a. Tento způsob pro základní prezentaci využívá např. [21] nebo i [5]. Metoda je velmi jednoduchá a především rychlá. Nevýhodou je, že je velmi vzdálena realitě a dává pouze základní přehled o pohybu simulované látky.

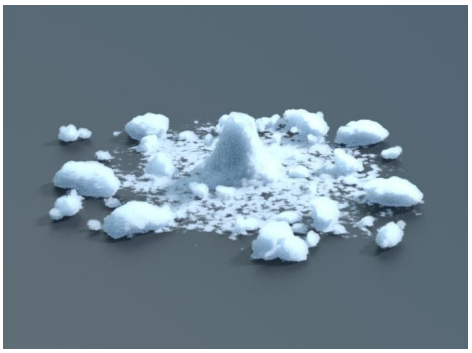
Pokročilejší technikou, která již poskytuje reprezentaci povrchu kapaliny viz 3.17b, je algoritmus marching cubes, který k tomuto účelu využívá opět



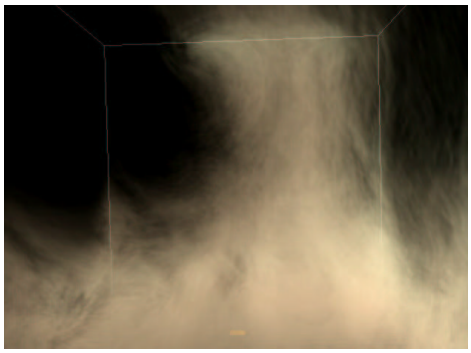
(a): Vykreslení zástupných objektů na pozicích částic, převzato z [26].



(b): Vykreslení výsledků simulace s využitím algoritmu marching cubes k získání povrchu simulované kapaliny, převzato z [16].



(c): Vykreslení sněhové koule po dopadu na zem s využitím volumetrického path-tracingu zachycující rozptyl světla, převzato z [32].



(d): Kouř vykreslený s využitím volumetrického renderingu a prolínání 3D textur, převzato z [14].

Obrázek 3.17: Různé způsoby vizualizace výsledků fluidní simulace.

např. [21]. Algoritmus však k dosažení výsledku využívá mřížkové struktury, při použití s Lagrangeovskými částicemi je tak třeba potřeba tuto strukturu vytvořit. V případě Newtonovských metod tento problém odpadá a je možné využít přímo simulační mřížku. Marching cubes provádí triangulaci v buňkách mřížky na základě hodnot v jejích vrcholech. Hodnoty jsou porovnávány s parametrem, který rozděluje prostor na část vně simulované látky a na simulovanou látku samotnou. Dle získané konfigurace je pak každé buňce mřížky přiřazena konkrétní triangulace. Problémem této metody je, že jejím výsledkem je pouze povrch simulované látky, je tedy nevhodná pro látky průhledné, zejména pak plyny, pokud není zkombinována s další metodou, jakou je např. ray-tracing.

Další oblíbenou metodou je ray-tracing implicitních povrchů [33]. V této metodě je simulované těleso popsáno množinou implicitních funkcí spolu se specifikací jejich vzájemné kombinace. Implicitní funkce je určena rovnicí, která každému bodu v prostoru přiřazuje určitou hodnotu. Tuto hodnotu můžeme považovat například za hustotu tělesa či intenzitu síly v daném místě

prostoru. V rovnici popisující implicitní plochu vystupují implicitní funkce, jejichž hodnota je závislá na poloze bodu. Při vlastním renderingu je využito hraniční hodnoty, od které je těleso vykreslováno [34].

Je-li ve fázi vizualizace použit ray-tracing, je možné obohatit výsledek za pomoci pokročilejších metod volumetrické vizualizace, které již umožňují zachytit i pokročilé vizuální vlastnosti látek, viz 3.17c. Např. práce [35] se zaměřuje na rendering sněhu s využitím implicitních povrchů (zde konkrétně metaballs), ale zejména do vizualizace zahrnuje rozptyl světla. Výsledku je dosaženo sledováním paprsku skrz voxely mřížky a aplikací fázové funkce na paprsek, která určí míru rozptylu. Podobné vizualizační metody je využito i při renderingu výsledků práce [32], která byla prezentována v sekci 3.3.3 věnovné MPM. Zde je výhodně využita po vizualizaci přímo simulační mřížka. Pokud by měl být ray-tracing s rozptylem světla využit při vizualizaci Lagrangeovského částicového systému, je nejprve nutné vytvořit příslušnou voxelovou strukturu, skrz kterou může být sledování provedeno.

Kapitola 4

Analýza a návrh

Z poznatků představených v předcházejících kapitolách vyplývají základní požadavky, které by měla splňovat výsledná aplikace vytvořená v rámci této práce.

- Aplikace by měla umožnit simulovat laviny kombinovaného typu, které se skládají jak z kompaktního, tak prachového sněhu.
- Aplikace by měla simulovat pohyb laviny v libovolném terénu, který je aplikací načten. Navíc, velikost laviny by neměla být omezena prostorově (pokud je k dispozici dostatek výpočetního prostoru).
- Uživatel by měl mít možnost ovlivnit parametry simulace, počáteční pozici laviny v terénu, vlastnosti sněhu apod.
- Aplikace by měla výsledky simulace prezentovat v jednoduché, ale dostatečně srozumitelné vizualizaci.

4.1 Zvolené řešení

Jako nejvhodnější k simulaci lavin kombinovaného typu se z metod uvedených v předcházející kapitole jeví metoda představená v [5], která k tekoucí části laviny využívá SPH a k simulaci prachového sněhu Newtonovskou mřížku. Tato metoda by měla poskytnout dostatečně věrnou simulaci, proto byla zvolena jako základ vznikající aplikace.

Při pohledu na průběh výpočtu obou složek uvedené metody je zřejmé, že se jedná o výpočetně náročné operace. Zároveň jsou však obě součásti, jak SPH, tak mřížková metoda, velmi dobře paralelizovatelné. Z tohoto důvodu bylo při výběru technologie pro navrhovanou aplikaci zváženo i toto řešení. Vzhledem ke schopnostem paralelních výpočtů na GPU bylo přistoupeno k využití technologie CUDA spolu s jádrem aplikace vytvořeným v jazyce C++. K vizualizaci byla zvolena knihovna OpenGL v kombinaci se shader programy napsanými v jazyce GLSL a k tvorbě uživatelského rozhraní pak knihovna Qt, která poskytuje dostatečně robustní řešení.

Vzhledem ke kombinaci Qt + CUDA bylo dále nutné přistoupit k využití externího nástroje pro automatizaci překladu CMake. Přestože Qt poskytuje svůj vlastní nástroj pro překlad, veškerý kód musí být přeložen jedním

kompilátorem, což v případě technologie CUDA představuje problém, jelikož CUDA využívá vlastní, se standardním C++, neslučitelný kompilátor.

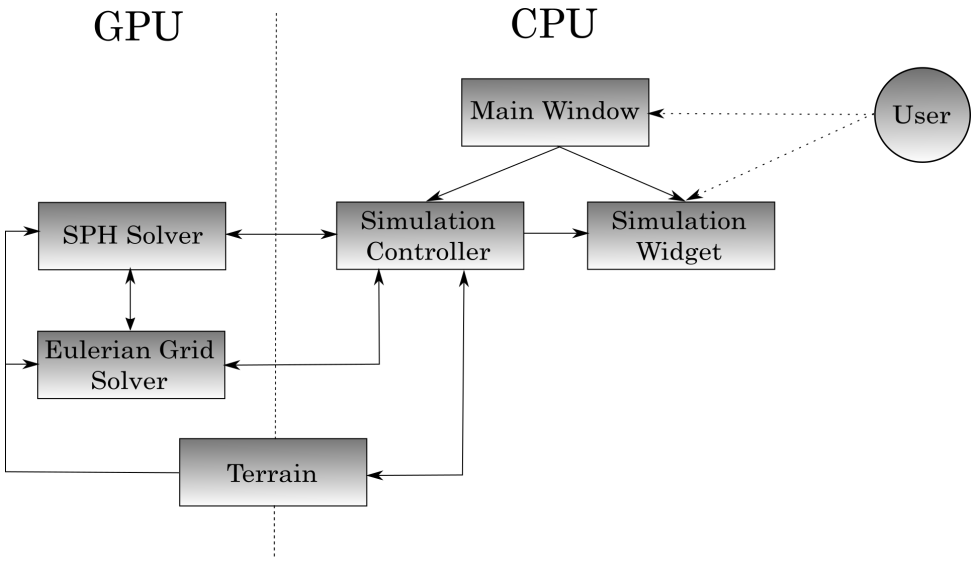
4.2 Návrh aplikace

Návrh aplikace spočívá v rozdělení do několika základních celků se specifickými funkcemi. Rozložení a provázání prvků je naznačeno na obrázku 4.1, následuje popis jednotlivých součástí:

- **Main Window** rozšiřuje třídu `QMainWindow` knihovny Qt a představuje hlavní okno aplikace. Stará se o vytvoření uživatelského rozhraní a propojení signálů akcí uživatele (např. stisknutí tlačítka) s metodami, které vykonávají příslušné akce v aplikaci.
- **Simulation Widget** rozšiřuje třídu `QOpenGLWidget` knihovny Qt. Třída reprezentuje okno prohlížeče a přímo pracuje s prvky knihovny OpenGL. Simulation Widget se stará o hlavní vykreslování scény, stav scény a o obsluhu uživatelských akcí v okně prohlížeče (např. tah myši pro rotaci). V okně reprezentovaném touto třídou jsou prezentovány veškeré výsledky simulace.
- **Simulation Controller** představuje jádro návrhu. Tato třída zajišťuje veškerou obsluhu simulačního procesu ze strany CPU, od obsluhy uživatelského nastavení přes kontrolu průběhu simulace po komunikaci s procesy probíhajícími na straně GPU.
- **Terrain** představuje terén, v němž probíhá simulace, načtený aplikací. Jelikož je terén načítán na straně CPU, ale musí být zároveň přístupný na GPU, je na obrázku 4.1 naznačen na rozhraní obou logických jednotek.
- **SPH Solver** obstarává část simulačního kroku týkající se tekoucí části laviny. Na konci každého simulačního kroku je výstupem pole částic na nových pozicích upravených dle interakčních sil. SPH Solver zároveň obstarává i simulaci akumulované vrstvy sněhu. Částice, které nejsou lavinou strženy jsou taktéž součástí výstupu, ale jejich pozice se mezi simulačními kroky nemění.
- **Eulerian Grid Solver** obstarává druhou část simulačního kroku probíhající na mřížce a simuluje prachový sníh zvířený pohybem laviny. Tato část je kriticky závislá na výstupu SPH Solveru, neboť částice z tekoucí vrstvy jsou jediným zdrojem hustot přítomným v simulaci. Výstupem této části jsou pozice buněk mřížky a hustoty v nich obsažené.

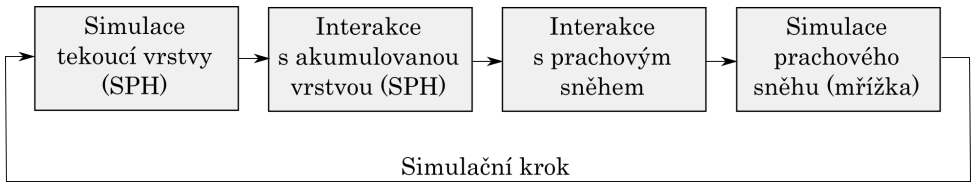
4.2.1 Simulace na GPU

Simulační krok je na GPU rozdělen do dvou základních bloků, SPH části a části mřížkové. Výpočet na mřížce je natolik oddělen, že nemusí být v simulaci vůbec zahrnut a celý proces pak sestává pouze ze simulace samotných částic



Obrázek 4.1: Schéma interakce mezi součástmi aplikace, mezi vrstvou na GPU a na CPU a mezi uživatelem.

SPH. Výpočet tak může být částečně urychlen za cenu vynechání simulace prachového sněhu a jeho zpětné interakce s tekoucí vrstvou laviny. Opačný proces, při kterém by byla vynechána simulace tekoucí vrstvy a simulován byl pouze prachový sníh, umožněn není, neboť primární zdroje vstupující do mřížkové metody odpovídají výstupu simulace SPH. Celý simulační krok ve výchozím nastavení odpovídá schématu znázorněnému na obrázku 4.2.



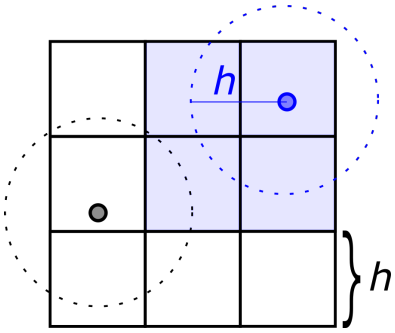
Obrázek 4.2: Schéma simulačního kroku na GPU.

Simulace tekoucí vrstvy

Návrh simulace tekoucí vrstvy na GPU počítá s přidělením výpočetního vlákna každé částici. Vlákno pak určuje nové hodnoty v bodě dané částice z částic sousedních: hustotu dle vzorce 3.23, a interakční síly z rovnic 3.27 a 3.32. V jednotlivých krocích jsou používány hodnoty již dříve známé nebo vypočítané. Je pouze nutné zajistit synchronizaci mezi jednotlivými kroky (např. po výpočtu hustoty, neboť vypočítané hodnoty jsou následně potřeba při výpočtu tlakových sil; nebo po výpočtu sil, kdy je nutné aby do integrace dle časového kroku vstupovaly síly obsahující přírůstek všech sousedních částic).

Návrh zároveň počítá s využitím akcelerační mřížky, která slouží k rychlejšímu nalezení sousedních částic. Tato struktura byla navržena jako uniformní

mřížka s krychlovými buňkami o hraně délky h. Tato délka je minimální možná, neboť parametr h určuje v rámci kernel funkcí nejzazší vzdálenost, ve které na sebe mohou částice působit. Touto volbou je zajištěno, že částice v interakční vzdálenosti se mohou nacházet pouze v buňkách sousedních k buňce obsahující částici, pro kterou je prováděn výpočet. Při kroku simulace pak stačí určit buňku obsahující částici zpracovávanou, a sousední částice hledat v této a 26 sousedních buňkách. Tato situace je zjednodušeně ve 2D znázorněna na obrázku 4.3. Výstavba a plnění mřížky v rámci GPU však zároveň kromě očekávaného zrychlení přináší nové synchronizační problémy při implementaci.

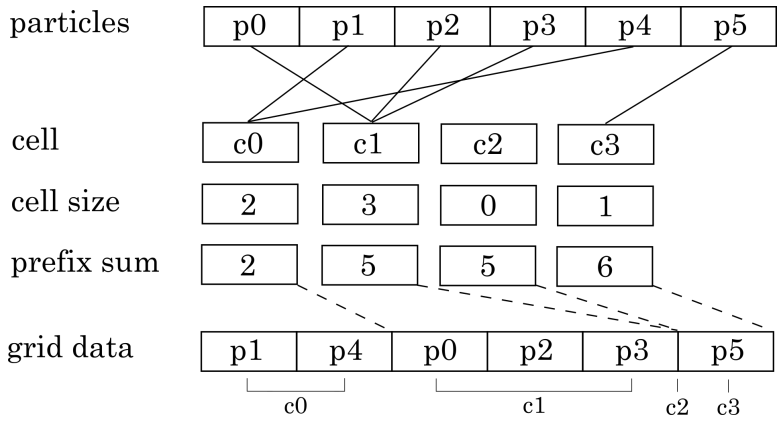


Obrázek 4.3: Rozměr buňky akcelerační mřížky je závislý na volbě parametru h. Díky této voblě se sousední částice modré částice mohou nacházet pouze v modrých buňkách mřížky.

Hlavním úskalím využití akcelerační mřížky je nutnost přiřazení každé částice buňce, která částici obsahuje. Pokud bychom zvolili přístup, kdy jsou zpracovávány buňky mřížky tak, že hledáme všechny částice, které se nachází v prostoru buňky, bude výpočet značně neefektivní. Pro každou buňku mřížky by totiž docházelo k procházení všech částic. Proto je potřeba přístup opačný: při zpracování jedné částice je určena buňka, ve které se částice nachází, a této buňce je přiřazen záznam o částici.

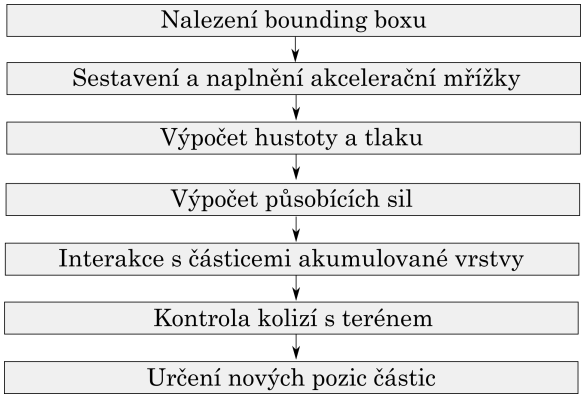
Vzhledem k tomu, že každá buňka akcelerační mřížky obsahuje dopředu neznámý počet částic, není vhodné použít buňky s fixním, dopředu alokovaným prostorem pro záznamy o částicích nacházejících se v buňce. V takovém případě by bylo mnoho prostoru nevyužito (buňka neobsahuje žádné částice) nebo by v horším případě nebyl dostatek prostoru k uložení záznamů. Je proto nutné navrhnout přístup, který zajistí korektní řešení, které nebude využívat pro záznamy pole o fixní velikosti a zároveň nebude produkovat synchronizační chyby. Jako vhodná se jeví metoda využívající prefixového součtu k určení počátečního indexu buňky v lineárním poli (obr. 4.4). Nejprve je nutné určit počet částic nacházejících se v každé buňce. Tento počet určuje prostor, který je nutné vyhradit pro danou buňku v lineárním poli záznamů. Součet velikostí všech předcházejících buněk pak určuje pozici, od které je prostor vyhrazen - tento součet je možné určit s využitím paralelních prostředků.

Při konstrukci akcelerační mřížky je nutno také počítat s tím, že se částice mezi jednotlivými simulačními kroky pohybují a mohou se dostat za hranice mřížky, pokud není adekvátně zvětšena. Mřížka se proto konstruuje pouze



Obrázek 4.4: Schéma vytvoření akcelerační mřížkové struktury metodou prefixového součtu. Mřížka je reprezentována lineárním polem o velikosti dpovídající počtu částic. Záznamy o částicích jsou v poli přeskupeny tak, aby odpovídaly pozicím v buňkách mřížky.

v prostoru, který částice skutečně obsahuje, v kvádru konvexní obálky všech částic. Jelikož data v mřížce jsou využitelná pouze v jednom simulačním kroku, může být mřížka takto sestavena v novém prostoru zcela nezávisle na předchozích pozicích mřížky.



Obrázek 4.5: Podrobné schéma simulace tekoucí vrstvy (SPH).

Celkový podrobný průběh simulace tekoucí vrstvy je shrnut ve schématu 4.5. Uživatel bude mít možnost měnit klidovou hustotu sněhu a koeficient viskozity.

Simulace vrstvy prachového sněhu

Simulace prachového sněhu probíhající na Eulerovské mřížce bude do značné míry závislá na výstupu simulace tekoucí vrstvy. Kupříkladu reakční síly k $\mathbf{f}_{i,j}^{drag}$ a $\mathbf{f}_{i,j}^{lift}$ z rovnic 3.46 a 3.47, které tvoří hlavní hnací sílu vstupující do vektorového pole, závisí na rychlostí částic. Obdobně mezi částicemi a mřížkou probíhá výměna určitého množství sněhu, přičemž část je přenesena z částic na mřížku (generování prachového sněhu dle rovnice 3.44) a část nao-

pak z mřížky na částici (adheze prachového sněhu k tekoucí vrstvě dle rovnice 3.45). Sníh přenesený z částic na mřížku odpovídá jedinému zdroji hustoty v simulaci.

Obrázek 4.6: Podrobné schéma simulace na mřížce.

Simulaci zároveň ještě dle rovnice 3.48 ovlivňují další síly, jmenovitě gravitační a vztlaková síla a síla $\mathbf{f}^{conf}$ vyplývající z metody vorticity confinement. Výpočet těchto sil není závislý na částicích a může probíhat separátně. Metoda vorticity confinement pracuje s hodnotami vířivosti v buňkách. Spočtené vířivosti jsou dále třeba při výpočtu interakční síly $\mathbf{f}_{i,j}^{lift}$, vyplatí se je tedy ukládat v samostatném poli. Průběh simulace na mřížce s dílčmi kroky je znázorněn na obrázku 4.6.

Výpočet rychlostního pole a pole hustot pak probíhá dle schématu popsaného v sekci 3.1.3, respektive 3.1.2. Jediným podstatným rozdílem je využití advekčního schématu vyššího řádu. Zvoleno bylo MacCormackovo schéma, použité např. v [36, 37]. Díky schématu vyššího řádu je v simulaci zachováno větší množství detailů. Průběh MacCormackovy advekce je popsán rovnicemi 4.1 až 4.3:

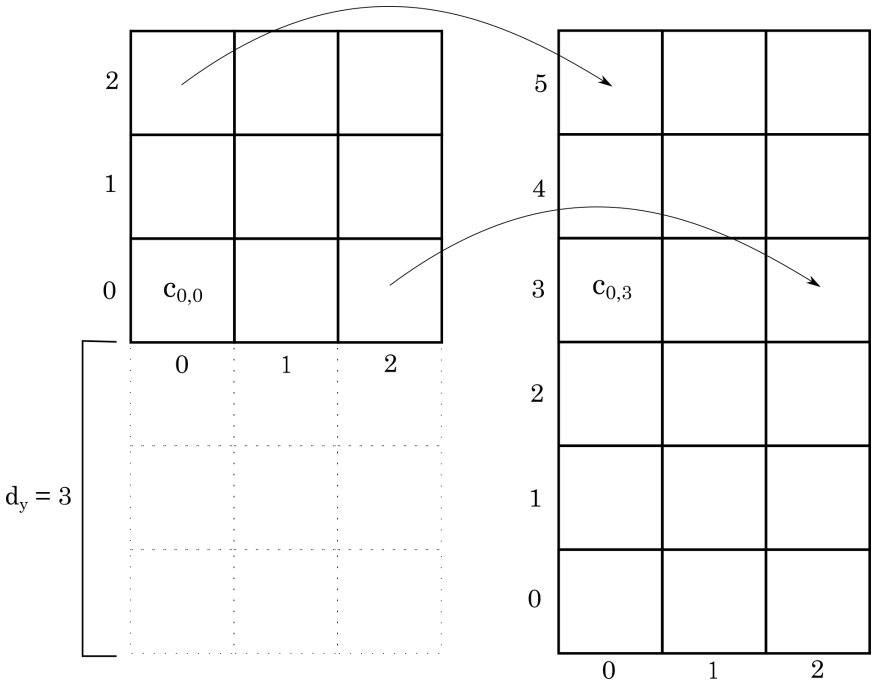
$$\hat{\phi}^{n+1} = A(\phi^n), \tag{4.1}$$

$$\hat{\phi}^{n} = A^R(\hat{\phi}^{n+1}), \tag{4.2}$$

$$\phi^{n+1} = \hat{\phi}^{n+1} + \frac{1}{2}(\phi^n - \hat{\phi}^n), \tag{4.3}$$

kde ϕ^n je hodnota vstupující do advekce a ϕ^{n+1} je hledaná hodnota po advekci. Pokud bychom za výsledek považovali rovnou výstup rovnice 4.1, bude výpočet odpovídat přímé advekci představené v sekci 3.1.3, zde A. MacCormackovo schéma se však snaží výsledek zpřesnit, proto je provedena zpětná advekce (časový krok je otočen) A^R. Rozdíl mezi hodnotami na počáteční pozici a na pozici získané zpětnou advekcí odpovídá zjištěné chybě, kterou je výsledek přímé advekce korigován.

Obdobně jako akcelerační mřížka nad částicemi tekoucí vrstvy musí i mřížka prachového sněhu reagovat na změny rozměrů. Zde je však situace poněkud komplikovanější, neboť hodnoty mřížky v aktuálním simulačním kroku vycházejí z hodnot simulačního kroku předcházejícího a navíc se musí po zvětšení nacházet v prostoru na totožném místě jako před zvětšením. Kvádr rozdělený



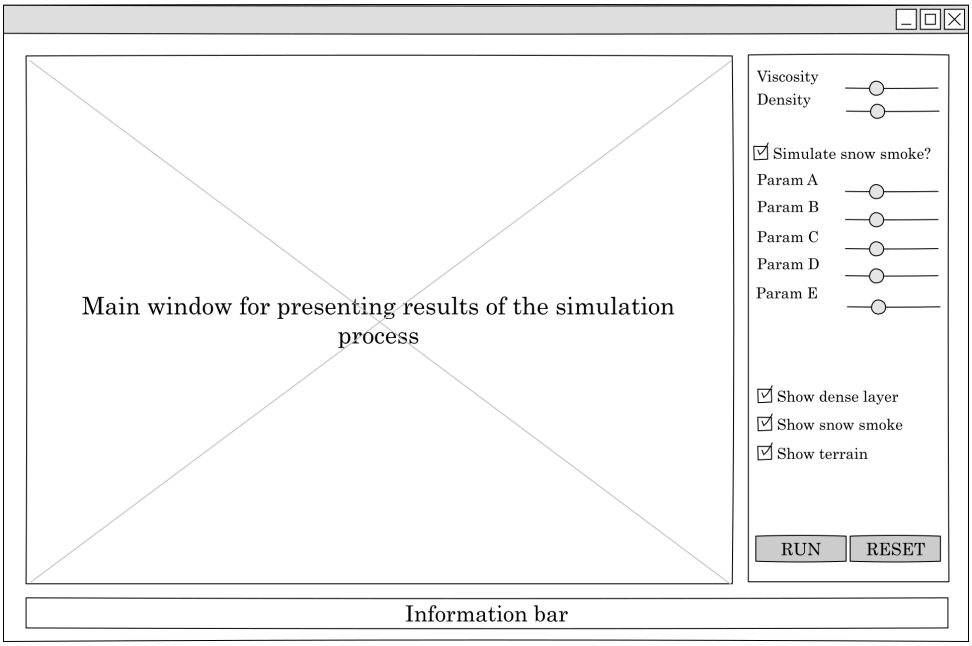
Obrázek 4.7: Ukázka reakce 2D mřížky na zvětšení. Indexace musí být upravena a staré hodnoty překopírovány do části mřížky, která odpovídá původnímu prostoru.

mřížkou se může zvětšit do 6 směrů v závislosti na chování simulace. Je potřeba adekvátně zareagovat a hodnoty předcházející menší mřížky umístit na správné pozice v mřížce zvětšené. Toho může být dosaženo poměrně jednoduchou přeindexací, kdy je nový index v rámci mřížky určen posunem původního indexu v každé dimenzi o velikost nové části mřížky. Tato situace je znázorněna pro mřížku zvětšenou ve směru osy y na obrázku 4.7. Nutno poznamenat, že posun bude nulový, pokud se nová část mřížky nachází ve směru vyšších souřadnic od mřížky původní.

4.3 Uživatelské rozhraní

Návrh uživatelského rozhraní aplikace byl vytvořen v nástroji Pencil a je prezentován na obrázku 4.8. Při návrhu byl zohledňován primárně požadavek na snadnou manipulaci se simulačním prostorem a průběhem simulace. Převážnou čás plochy aplikačního okna tak tvoří prostor vyhrazený vizualizaci. V tomto okně jsou jednak prezentovány výsledky simulace, ale zároveň zde probíhá významná část interakce se simulací, především pohyb 3D scénou (rotace, translace, přiblížení nebo oddálení), ale také výběr iniciálních souřadnic v terénu, odkud se má lavina šířit.

Druhou neméně podstatnou součástí uživatelského rozhraní tvoří ovládací panel umístěný vpravo od vykreslovacího okna. Zde se nachází veškeré prvky ovládající nastavení a průběh simulace. Pro číselné hodnoty v omezeném



Obrázek 4.8: Návrh uživatelského rozhraní vytvoření v aplikaci Pencil.

rozsahu (hustota, viskozita sněhu) jsou zvoleny posuvníky, které umožní pohodlnou změnu hodnoty. Spuštění nebo zastavení simulace je řízeno tlačítkem, obdobně jako resetování simulace do počáteční konfigurace. Pro lepší orientaci ve scéně a v prezentovaných výsledcích je umožněno skrýt jednotlivé nezávislé složky simulace (tekoucí část laviny tvořená částicemi, prachový sníh na mřížce, terén).

4.4 Vizualizace

Vizualizace výsledků simulace je dle návrhu realizována pomocí OpenGL a shader programů v jazyce napsaných GLSL. Nejedná se o stěžejní část práce, proto je u tekoucí vrstvy zvolena základní metoda vykreslování zástupných objektů (koule) na pozicích částic. Aby byla demonstrována interakce s vrstvou prachového sněhu, je hmotnost částic namapována na barvu. Obdobně je navržena i vizualizace prachového sněhu, kdy jsou zástupné objekty vykresleny v bodech odpovídajících příslušným buňkám mřížkové struktury. Prachový sníh je zobrazován s průhledností závisející na množství sněhu obsaženého ve vykreslované buňce. Návrh zároveň počítá s možností zobrazení rychlostního pole příslušejícího k mřížce. Pro přehlednost je umožněno všechny tři zmíněné součásti simulace a dále terén zobrazit samostatně nebo zároveň.

Kapitola 5
Implementace

Implementace návrhu probíhala v několika etapách. Prvně bylo nezbytné vytvořit základní rozhraní aplikace s vizualizačním oknem, ve kterém by bylo možné kontrolovat výstup produkovaný simulačním procesem. Zároveň bylo potřeba již v začátcích připravit načítání terénu, jelikož je nedílnou součástí simulace lavin. Konečně následovala implementace dvou hlavních částí aplikace, řešiče SPH a mřížkové metody. Jako první byla logicky implementována metoda SPH, neboť na jejím výstupu je dle návrhu závislá mřížková část. Částicová metoda SPH byla nejprve vyzkoušena na CPU, ale po zhodnocení nízké výpočetní efektivity, viz kapitola 6, byla implementace na CPU opuštěna a dále probíhala výhradně na GPU s využitím technologie CUDA.

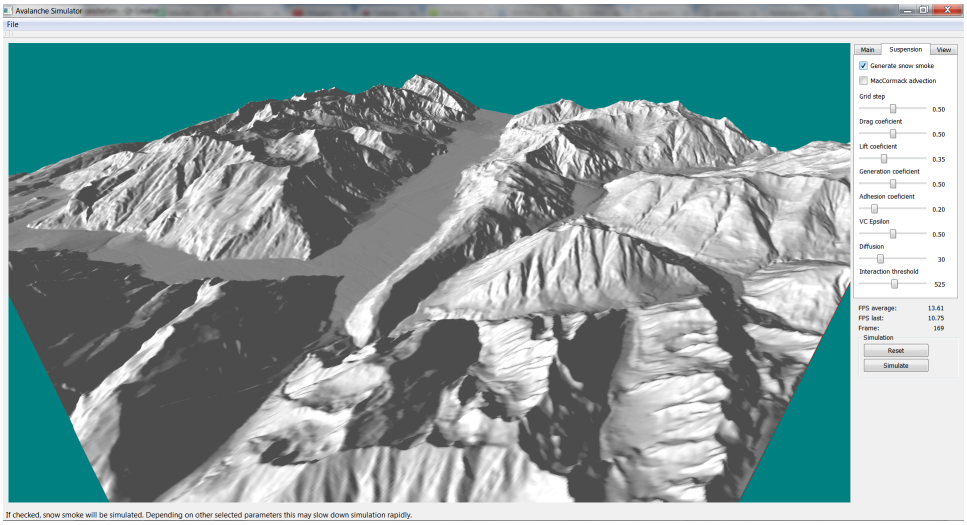
Obě hlavní části simulace potřebují ve svém průběhu přístup k informacím o prostoru, ve kterém simulace probíhá. Ať už se jedná o akcelerační mřížku v případě SPH nebo základní mřížku v simulaci prachového sněhu, vždy je potřeba mít přístup k údajům o prostoru, který je mřížkou vymezen. Proto byla zavedena struktura `gridParams`, viz kód 5.1, která tyto základní parametry poskytuje a zároveň je možné ji jednoduše sdílet. Struktura sice používá CUDA vektorové typy (`float3`, `int3`), ty je ovšem možno používat i na straně CPU a mřížkové parametry tak mohou být sdíleny i mezi GPU a CPU.

```
struct gridParams{
float3 minC;     ///< Minimal coordinates of grid.
float3 maxC;     ///< Maximal coordinates of grid.
int3 dim;        ///< Grid dimensions.
int maxIndex;    ///< Maximal grid array index.
};
```

Ukázka kódu 5.1: Struktura `gridParams`.

5.1 Terén

Kromě výchozího generovaného terénu aplikace umožňuje načtení výškových dat, ze kterých je terén složen. Vzhledem ke snadné dostupnosti a širokému pokrytí reálných lokalit byla zvolena data SRTM (Shuttle Radar Topography Mission) ve formátu HGT, dostupná např. z [38].



Obrázek 5.1: Výsledná aplikace s načteným terénem. V pravé části je vidět panel s nastavením simulace vrstvy prachového sněhu.

Data představují pouze výškové hodnoty odpovídající bodům v horizontálně orientované 2D mřížce. Aby mohl být vykreslen terén jako celek, musí být nad daty provedena triangulace. Vzhledem k uniformnímu a dopředu známému rozdělení mřížky terénu je triangulace přímočará: mezi každými čtyřmi sousedními buňkami vziknou dva trojúhelníky. Pro kontrolu kolizí s terénem a pro vizualizační účely je dále nezbytné znát normály ve vrcholech trojúhelníků. Proto jsou nejprve určeny normálové vektory pro každý trojúhelník získaný triangulací a následně jsou interpolací ze sousedních trojúhelníků určeny normály ve vrcholech, což je nezbytné pro dosažení hladkého stínování při vykreslování i pro lepší kontrolu kolizí.

Trojúhelníky je naplněn buffer, který je posílán do vykreslovacího řetězce. V souvislosti s terénem jsou použity dvě dvojice shader programů. Základní `vShaderTerrain.vert` a `fShaderTerrain.frag` slouží k přímému vykreslení terénu v okně prohlížeče, přičemž je aplikováno ambientní a difuzní osvětlení. Druhou dvojici tvoří `vShaderPicking.vert` a `fShaderPicking.frag`, jejichž úkolem je vykreslit scénu do uživateli neviditelného offscreen bufferu. Tyto shader programy jsou použity pouze tehdy, má-li být realizován výběr iniciálních souřadnic pro umístění laviny kliknutím na terén. Každé dva trojúhelníky jsou namapovány na jedinečnou barvu a scéna je takto vykreslena. Dle barvy pixelu v bodě kliknutí může být zpětně určen vybraný trojúhelník a tedy i odpovídající pozice nad terénem. Tato metoda, známá také jako color picking, je hlouběji popsána např. v [39, 40].

```
tex2D(terrainRef, position.y+0.5f, position.x+0.5f);
```

Ukázka kódu 5.2: Určení výšky terénu v bodě částice.

Na GPU jsou poslány pouze původní výškové hodnoty v bodech mřížky a normály těmto bodům odpovídající. Obě pole jsou namapovány na CUDA texturu. Využití textur pro terén přináší i podstatné ulehčení implementace

kolizí, neboť hodnoty mezi vrcholy jsou automaticky interpolovány. Díky této vlastnosti je pak určení výšky terénu pod/nad částicí otázkou jediného volání, viz 5.2.

5.2 Vrstva kompaktního sněhu

Simulace vrstvy kompaktního sněhu (metoda SPH) je realizována v souboru `gpuParticles.cu`. Hlavní část dat představující částice (pozice, hustota, tlak, hmotnost, rychlost, síla) je uložena v globální paměti, každá složka v samostatném poli. Do sdílené paměti jsou částicová data z paměti globální kopírována pouze v případě náročných operací (prexiový součet, paralelní redukce). Konstanty (předpočítané hodnoty vyhlazovacích funkcí, hodnoty parametrů N, h, ...) jsou uloženy v rychlé konstantní paměti.

Výpočet realizuje několik CUDA kernel funkcí, členění vychází zejména z potřeby synchronizace mezi jednotlivými kroky. Synchronizace musí být vynucena nad všemi výpočetními vlákny zároveň, neboť používané hodnoty se mohou nacházet na libovolné pozici v paměti. Výpočet je proto rozdělen na jednotlivé úseky, mezi nimiž je vynucena synchronizace voláním `cudaDeviceSynchronize()`. Stěžejní kernely pracující přímo s částicemi jsou spouštěny na počtu jader, které odpovídá počtu částic, počet jader určený pro kernely stavějící akcelerační mřížku odpovídá počtu buněk mřížky a paralelní redukce je spouštěna na fixním počtu jader v rámci jednoho bloku (sice jedno vlákno redukuje větší počet hodnot, ale odpadá komplexnější synchronizace).

Pokud by nebyla použita akcelerační mřížka, stačily by k výpočtu tři kernely: `baseKernel`, který nastaví pole hustot na výchozí hodnoty odpovídající vlastní hustotě částice, `densPressKernel`, ve kterém dochází k výpočtu přírůstku hustoty z okolních částic a určení tlaku v bodě částice a `forceKernel`, který dokončí krok simulace výpočtem sil působících na částici a integrací pozice dle časového kroku.

Vzhledem k potřebě vyšší stability simulace je do výpočtu dále zařazen kernel `updateTimeStep`, který umožňuje zkrátit délku časového kroku simulace vzhledem k nejvyšší rychlosti částice vyskytující se v simulaci. Díky této úpravě se během jednoho kroku simulace nemůže částice posunout po dráze delší než h. Výpočet je proveden paralelní redukcí nad polem obsahujícím velikosti rychlostí.

Nejzajímavější částí implementace na GPU je pak rozšíření o akcelerační strukturu. Dle poznatků uvedených v kapitole 4 musí mřížka rozdělovat pouze prostor, který částice skutečně zaujímají a musí reagovat na jejich pohyb svou velikostí. Z uvedených nároků je zřejmé, že není možné mřížku sestavit pouze jednou na začátku simulace, ale že musí být postavena v jejím každém kroku na základě aktuálních pozic všech částic. Aby bylo možné určit rozměr mřížky, je potřeba nejprve získat minimální a maximální souřadnice, na kterých se vyskytují částice. K tomuto účelu slouží kernel `updateBoxDimensions`, který provede paralelní redukci nad pozicemi částic a uloží nalezené extrémy. Tento kernel může být spouštěn zároveň s kernelem `updateTimeStep`, není mezi nimi nutná synchronizace.

Jakmile jsou na začátku simulačního kroku získány údaje o aktuálním rozměru mřížky, je možné začít s její výstavbou. Výstavba mřížky je založena na metodě paralelního prefixového součtu. Dochází k úpravě kernelu `baseKernel`, který nyní navíc obsahuje výpočet indexu buňky, ve kterém se částice nachází, zvýšení počtu částic v této buňce o jedna a uložení indexu částice v rámci buňky. Průběh základního kernelu je znázorněn pseudokódem 1.

Pseudokód 1 Průběh kernelu `baseKernel`.

```
1: procedure BASEKERNEL
2:     idx ← index of particle
3:     density[idx] ← self density
4:     gridIdx ← GETGRIDINDEX(position[idx])
5:     cellIndices[idx] ← cellsSize[gridIdx]   //atomic
6:     cellsSize[gridIdx] ← cellsSize[gridIdx] + 1   //atomic
```

Následně je proveden paralelní prefixový součet nad počty částic v buňkách, čímž je získán počáteční index každé buňky mřížky. Samotný výpočet musí vzhledem k dopředu neznámé velikosti mřížky probíhat ve dvou úrovních, nejprve v rámci bloků a následně nad bloky. K tomuto účelu slouží dva nové kernely, `buildGridKernel` a `blockSumKernel`. Nakonec je nově rozvržená mřížka naplněna indexy do polí částic, viz ukázka kódu 5.3.

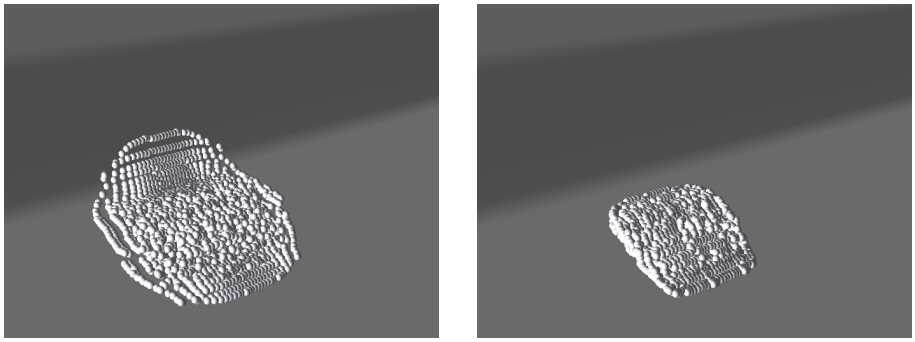
```
__global__ void fillGridKernel(...){
int idx = blockIdx.x*blockDim.x + threadIdx.x;
if(idx >= N) return;

int gridIdx = getGridIdx(positions[idx], grid);
int posInPreSum = gridHelper[gridIdx];
gridData[posInPreSum + cellIndices[idx]] = idx;
cellIndices[idx] = gridIdx;
}
```

Ukázka kódu 5.3: Naplnění mřížky na základě hodnot prefixového součtu v kernelu fillGridKernel.

Stěžejní část simulace realizují kernely `densPressKernel` a `forceKernel` naznačené pseudokódy 2 a 3. Kernel `densPressKernel` počítá přírůstek hustoty z okolních částic dle rovnice 3.23 a určí tlak na pozici částice dle rovnice 3.28. Kernely musí být odděleny synchronizací, neboť při výpočtu sil v kernelu `forceKernel` je potřeba znát aktuální hodnoty hustoty a tlaku. Zvolená referenční klidová hustota použitá při výpočtu tlaku má vliv na výsledek simulace, neboť částice se v prostoru snaží dosáhnout rozložení, které této hodnotě odpovídá. Vliv velikosti klidové hustoty je znázorněn na obrázku 5.2.

Kernel `forceKernel` nejprve provede výpočet sil působících na částici z částic sousedních dle rovnic 3.27 (tlaková síla) a 3.32 (viskozita). Velikost parametru viskozity μ má značný vliv na chování simulace. S rostoucí hodnotou parametru jsou částice více ovlivněny pohybem částic okolních a pohyb kapaliny je vyhlazen. Vliv viskozity na simulaci je znázorněn na obrázku 5.3. Nakonec je provedena integrace dle časového kroku a výpočet je zakončen



(a): $\rho_0 = 100\,\mathrm{kg/m^3}$ **(b):** $\rho_0 = 1000\,\mathrm{kg/m^3}$

Obrázek 5.2: Vliv velikosti klidové hustoty na průběh simulace při jinak shodných parametrech (včetně stejné hmotnosti) a ve stejném simulačním kroku. Částice byly spuštěny po nakloněné rovině s proměnným sklonem.

Pseudokód 2 Průběh kernelu `densPressKernel`.

```
1: procedure densPressKernel
2:     idx ← index of particle
3:     gridIdx ← cellIndices[idx]
4:     for each neighbouring cell do
5:         for each particle in cell do
6:             idx2 ← index of particle using prefix sum
7:             if particles on idx and idx2 close enough then
8:                 density[idx] ← density[idx] + contribution of particle idx2
9:     pressure[idx] ← computed pressure
```

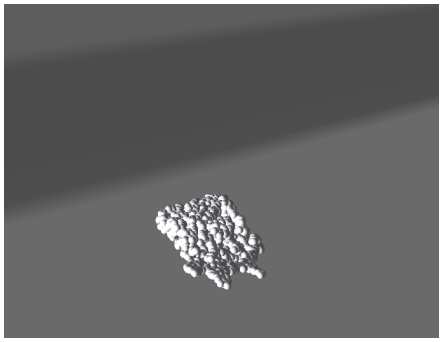
kontrolou kolizí s terénem, viz ukázka kódu 5.4.

```
__global__ void forceKernel(...){
...
float3 acceleration = forces[idx] / densities[idx];
acceleration.z -= 9.81f; // Apply gravity
velocities[idx] += acceleration * *timeStep;
positions[idx] += velocities[idx] * *timeStep;
checkCollisions(positions[idx], velocities[idx], timeStep);
}
```

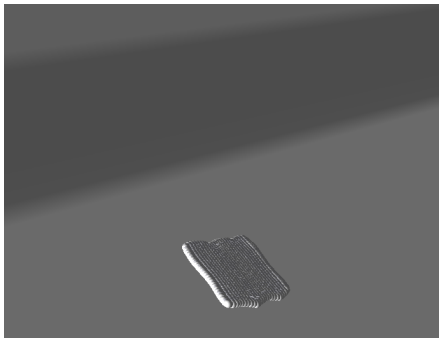
Ukázka kódu 5.4: Integrace dle časového kroku v kernelu forceKernel.

5.3 Vrstva prachového sněhu

Simulace vrstvy prachového sněhu (mřížková metoda) je realizována v souboru `gpuSuspension.cu`. Z popisu mřížkové metody v sekci 3.1 plyne, že pro každou buňku musí být uložena dvojice polí pro hustoty a pro rychlosti, jedno vždy odpovídající předcházejícímu kroku a jedno kroku aktuálnímu. Navíc, každá složka vektoru rychlosti je uložena v samostatném poli, v základu je tedy potřeba osm 1D polí. Jelikož je šíření prachového sněhu také omezeno



(a): $\mu = 1$



(b): $\mu = 200$

Obrázek 5.3: Vliv velikosti parametru viskozity na průběh simulace při jinak shodných parametrech (včetně stejné hmotnosti) a ve stejném simulačním kroku. Částice byly spuštěny po nakloněné rovině s proměnným sklonem.

Pseudokód 3 Průběh kernelu `forceKernel`.

```
1:  procedure forceKernel
2:      idx ← index of particle
3:      gridIdx ← cellIndices[idx]
4:      for each neighbouring cell do
5:          for each particle in cell do
6:              idx2 ← index of particle using prefix sum
7:              if particles on idx and idx2 close enough then
8:                  force[idx] ← force[idx] − computed pressure force
9:                  force[idx] ← force[idx] + computed viscosity force
10:     acceleration ← force[idx]/density[idx]
11:     acceleration ← gravitational force
12:     velocity[idx] ← velocity[idx] + acceleration ∗ timeStep
13:     position[idx] ← position[idx] + velocity[idx] ∗ timeStep
14:     checkCollisions(position[idx], velocity[idx], timeStep)
```

terénem, je zavedeno další pole k rozlišení buněk na *hraniční* a *mimo hranici*. Při výpočtu se také uplatňuje vířivost (vektor, tedy tři 1D pole) a její velikost (jedno 1D pole), celkem je tedy potřeba třináct 1D polí, jejichž délka odpovídá celkovému počtu buněk mřížky. Parametry simulace a další neměnné hodnoty, jako délka hrany buňky, jsou uloženy opět v konstantní paměti.

```
__device__ int IX(int x, int y, int z, gridParams &grid){
return (x + y * grid.dim.x + z * grid.dim.x * grid.dim.y);
}
```

Ukázka kódu 5.5: Určení 1D indexu v mřížce z 3D indexu daného čísly x, y a z.

Výpočet je spouštěn na počtu jader, který odpovídá počtu buněk mřížky. Jelikož je často potřeba zjistit hodnotu v sousední buňce, je zavedena funkce *IX*, která převede 3D indexaci na 1D, viz ukázka 5.5. Výpočet samotný neprobíhá v okrajových buňkách mřížky, kde se aplikují okrajové podmínky, není proto třeba při každém přístupu do sousedních buněk kontrolovat, že

je index validní, neboť bude vždy (maximálně se přečte hodnota z okrajové buňky).

Simulace prachového sněhu na mřížce je rozdělena do základních kroků, jak bylo naznačeno v návrhu na schématu 4.6. V kódu je pak toto schéma dodrženo a simulační krok je rozdělen na jednotlivé funkce, viz ukázka kódu 5.6.

```
processSuspension(...){
updateGridSize(sph_grid);
computeVorticities();
computeGridForces();
interactWithSPH(sph_params...);
velocityStep();
diffuseStep();
}
```

Ukázka kódu 5.6: Dělení simulace prachového sněhu na jednotlivé bloky.

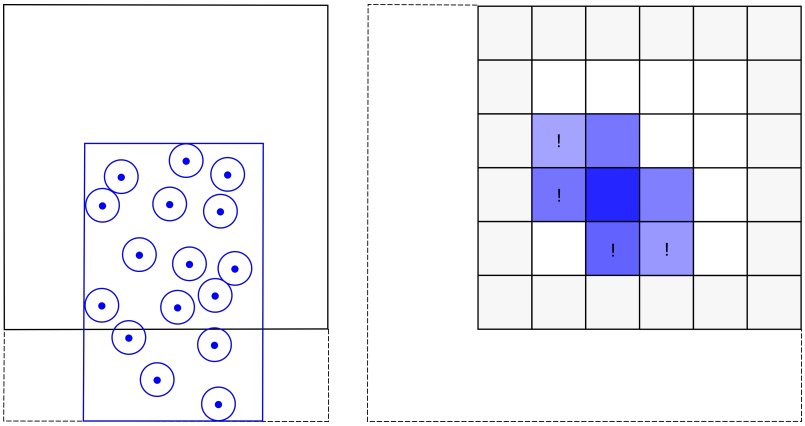
5.3.1 Zvětšení mřížky

V prvním kroku `updateGridSize(sph_grid)` je zajištěno zvětšování simulační mřížky v závislosti na průběhu simulace. Ke zvětšení mřížky může dojít ve dvou případech:

- Konvexní obálka SPH částic není celá obsažena v prostoru vymezeném simulační mřížkou, viz obrázek 5.4a (konvexní obálka částic *sph_grid* byla získána při aktualizaci akcelerační mřížky).
- Simulační mřížka obsahuje nenulové hodnoty hustoty v okrajových buňkách (ale nikoliv v buňkách hraničních, kde se uplatňují hraniční podmínky), viz obrázek 5.4b. Kontrolu hodnot provádí kernel *checkResizeKernel*, jehož výstupem je pouze indikace typu ano / ne. Z rychlostních důvodů byla nakonec kontrola nenulových hodnot nahrazena kontrolou na hodnoty přesahující kladnou nenulovou konstantu, neboť při kontrole nenulových hodnot docházelo k příliš častému zvětšování mřížky.

Pokud bylo zjištěno, že nastává alespoň jeden výše uvedený případ, mřížka bude zvětšena. Pokud nastanou oba případy zároveň, je uvažován extrém (mřížka bude zvětšena více). Je spočítán nový rozměr mřížky a ta je následně zvětšena zvlášť v každém směru, ve kterém má ke změně dojít. Navíc je nutné zajistit správnou přeindexaci hodnot nacházejících se v mřížce, což má na starost funkce `remap(gridParams *grid_new, int xAdd, int yAdd, int zAdd)`, která obdrží parametry nové mřížky a počet nových buněk v každém směru. Funkce zajistí realokaci všech CUDA polí, přičemž je využito faktu, že pro každou hodnotu (hustota, rychlost) existuje dvojice polí. Jedno z nich tak může být alokováno do nového rozměru, hodnoty druhého jsou do něj zkopírovány a teprve poté je zvětšeno i druhé pole (CUDA neumožňuje přímou realokaci se zachováním hodnot).

O přemapování indexů v každém poli se pak stará *remapKernel*, viz ukázka kódu 5.7. Jelikož původní metoda navržená Stamem [14] interně pracuje



(a): Zvětšení v důsledku SPH částic mimo prostor mřížky.

(b): Zvětšení v důsledku nenulových hodnot hustoty v krajních buňkách.

Obrázek 5.4: Případy, kdy může dojít ke zvětšení mřížky. Stará mřížka je ohraničena černou nepřerušovanou čarou, zvětšená mřížka je naznačena přerušovaně.

s mřížkou, jejíž velikost v každém směru je jednotková, musí být při zvětšení přepočítány i hodnoty rychlosti (hodnoty hustoty nejsou tímto ovlivněny). Toto přepočítání je určeno parametrem *scale*, který odpovídá počtu buněk v daném rozměru v nezvětšené mřížce děleném počtem buněk v daném rozměru v mřížce zvětšené.

```
arr_new[IX(x+xAdd, y+yAdd, z+zAdd, grid)] = scale*arr_old[idx];
```

Ukázka kódu 5.7: Přeindexování hodnot ze staré mřížky do mřížky nové v kernelu *remapKernel*.

Po každém zvětšení mřížky je potřeba určit nové hraniční buňky. Toho je docíleno voláním kernelu *setBoundariesKernel*, který označí všechny buňky nacházející se pod úrovní terénu jako hraniční.

5.3.2 Vířivost

V každém simulačním kroku jsou aktualizovány hodnoty vířivosti určené jako $\omega_j^{\mathbf{g}} = \nabla \times \mathbf{u}_j^{\mathbf{g}}$, kde $\mathbf{u}_j^{\mathbf{g}}$ je rychlost v buňce j mřížky. Tento výpočet provádí *vorticityKernel*, který zároveň spočítá a uloží délky vektorů vířivosti $|\omega_j^{\mathbf{g}}|$. Jakmile jsou hodnoty známy, může být rychlostní pole pro další simulační krok inicializováno silami $\mathbf{f}^{buoyancy}$ a $\mathbf{f}^{conf}$ z rovnice 3.48. Výpočet a aplikaci sil realizuje kernel *gridForceKernel*. Tyto síly vychází přímo z již známé konfigurace na mřížce a mohou být spočítány bez interakce s částicemi.

5.3.3 Výpočet pole rychlostí

Výpočet rychlostního pole je znázorněn pomocí pseudokódu 4 (jedná se o přímou realizaci návrhu 3.3). Jelikož program pracuje s poli, která každé zastupuje jednu složku vektoru, jsou i jednotlivé kroky výpočtu kromě projekce

realizovány samostatně pro každou dimenzi. Navíc je nutné po každé úpravě pole vynutit aplikaci hraničních podmínek.

Pseudokód 4 Pseudokód výpočtu rychlostního pole.

```
1:  procedure VELOCITYSTEP
2:      Add sources to vel_x, vel_y, vel_z arrays
3:      DIFFUSE(vel_x)
4:      DIFFUSE(vel_y)
5:      DIFFUSE(vel_z)
6:      PROJECT()
7:      ADVECT(vel_x)
8:      Enforce boundary on vel_x
9:      ADVECT(vel_y)
10:     Enforce boundary on vel_y
11:     ADVECT(vel_z)
12:     Enforce boundary on vel_z
13:     PROJECT()
```

Procedura `diffuse()` realizuje řešení rovnice 3.7 pomocí Gauss-Seidelovy relaxace. Ta je provedena opakováním stejného výpočtu nad daty konvergujícími s každou iterací k požadované hodnotě. Problém nastává na GPU, kde není možné relaxaci provést v rámci jednoho spuštění kernelu, neboť hodnoty v polích se mezi jednotlivými iteracemi mění a je potřeba zajistit, aby všechna vlákna měla přístup k aktuálním hodnotám. Proto je příslušný kernel spouštěn v cyklu, jehož délka odpovídá počtu iterací a mezi jednotlivými iteracemi je provedena synchronizace vláken, jak je naznačeno na pseudokódu 5.

Pseudokód 5 Pseudokód procedury realizující difuzi opakovaným voláním příslušného kernelu. Počet iterací vychází ze Stamova návrhu, viz. [28]

```
1:  procedure DIFFUSE(arr, arr_prev, dim)
2:      for i ∈ {1, ..., 20} do
3:          DIFFUSEKERNEL(arr, arr_prev)
4:          Synchronize
5:          Enforce boundary in dim dim
6:          Synchronize
```

Projekce je realizována procedurou `project()`. V projekci jsou použita dvě pomocná pole pojmenovaná `p` a `div` (fyzicky jsou použita dvě aktuálně nepoužívaná pole rychlosti). Nejprve jsou spočteny hodnoty divergence v buňkách (kernel `project1Kernel`). Následuje výpočet řešení Poissonovy rovnice, který odpovídá Gauss-Seidelově relaxaci použité při difuzi. Po tomto kroku odpovídá `p` poli gradientů, které jsou od rychlostního pole odečteny v kroku posledním (kernel `project3Kernel`).

Advekce je provedena v závislosti na zvoleném schématu buď kernelem `advectKernel` nebo `advectKernelMacCormack`. V jednodušším případě je

Pseudokód 6 Pseudokód procedury realizující projekci přes pomocná pole p a div (jsou využita volná pole z předcházejícího kroku).

```
1:  procedure PROJECT
2:      PROJECT1KERNEL(p, div, vel_x, vel_y, vel_z)
3:      Synchronize
4:      Enforce boundary on p
5:      Enforce boundary on div
6:      Synchronize
7:      PROJECT2 which equals DIFFUSE(p, div, 0)
8:      PROJECT3KERNEL(p, vel_x, vel_y, vel_z)
9:      Synchronize
10:     Enforce boundary on vel_x
11:     Enforce boundary on vel_y
12:     Enforce boundary on vel_z
13:     Synchronize
```

přečtena rychlost ve zpracovávané buňce, přetransformována na reálnou velikost dle rozměru mřížky, odečtena od pozice buňky k získání cílové pozice, v níž je trilineární interpolací získána nová hodnota.

Pokud je použito MacCormackovo schéma, je v cílové pozici získané způsobem výše také přečtena rychlost a tentokrát pro změnu k cílové pozici přičtena (návrat zpět). Rozdíl mezi hodnotou na pozici po přičtení zpětné rychlosti a hodnotou původní je použit jako korekce numerické chyby, viz rovnice 4.3.

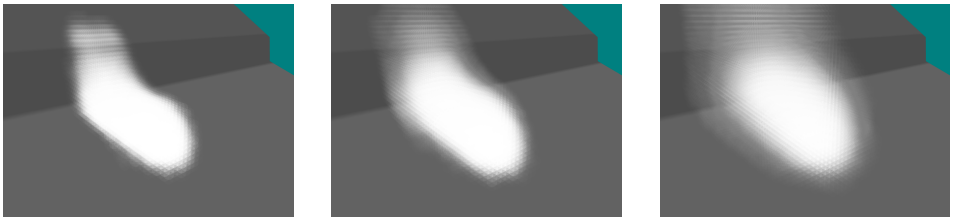
5.3.4 Výpočet pole hustot

Výpočet pole hustot je znázorněn pomocí pseudokódu 7. V této části jsou znovu využity procedury, které již byly popsány v rámci výpočtu rychlostního pole.

Pseudokód 7 Pseudokód výpočtu pole hustot.

```
1:  procedure DENSITYSTEP
2:      Add source to density array
3:      DIFFUSE(density)
4:      ADVECT(density)
5:      Enforce boundary on density
```

Drobný rozdíl se nachází pouze v difuzní části, do které vstupuje parametr ovlivňující míru difuze hustoty. Nastavením tohoto parametru je možné výrazně ovlivnit míru rozptylu prachového sněhu v lavině, jak je znázorněno na obrázku 5.5.



(a): $d = 1$ **(b):** $d = 20$ **(c):** $d = 100$

Obrázek 5.5: Vliv velikosti difuzního parametru d na průběh simulace. Zdroje do rychlostního pole i do pole hustot jsou generovány částicemi, které byly spuštěny po nakloněné rovině s proměnným sklonem. Pro názornost je zobrazen pouze terén (nakloněná rovina) a hodnoty pole hustot.

5.4 Interakce mezi vrstvami

Interakce mezi částicemi SPH a mřížkou probíhá v rámci jediné procedury `interactWithSPH()` poté, co již proběhla aktualizace rozměru mřížky a rychlostní pole byla inicializována silami $\mathbf{f}^{buoyancy}$ a $\mathbf{f}^{conf}$, viz ukázka kódu 5.6. Procedura obdrží veškerá potřebná data částic (pozice, síly, rychlosti, hmotnosti, poloměr částice a také objemovou hustotu) a spustí kernel `interactionKernel`. Tento kernel není spuštěn na počtu jader, které odpovídá buňkám mřížky, jak by mohlo naznačovat jeho umístění v bloku realizujícím simulaci na mřížce, ale naopak je spuštěn z pohledu částic. Tento přístup je zvolen proto, že je velmi snadné určit buňku mřížky, ve které se částice nachází, ale obtížnější nalézt všechny částice, které spadají do buňky (tento přístup by vedl přinejmenším na další použití paralelního prefixového součtu, což je v tomto případě zcela zbytečné).

```
__global__ void interactionKernel(...){
...
float3 f_drag = rel_vel * rel_vel_len *
     (sph_masses[idx] * -dragCoef ) / *sph_radius;
float3 f_lift = cross(rel_vel, Interpolated_vort) *
     sph_masses[idx] * -liftCoef;
float3 f_total = (f_drag + f_lift) * *timeStep;

float sectional_area = CUDART_PI_F * *sph_radius * *sph_radius;
float amount_generated = generationCoef * pow(rel_vel_len,1.7f)
    *
     4 * sectional_area * *timeStep;
float amount_adhered = adhesionCoef *
     density_prev[grid_linear_idx] * rel_vel_len *
     sectional_area * *timeStep;
float amount_total = amount_generated - amount_adhered;
...
}
```

Ukázka kódu 5.8: Výpočet v rámci kernelu `interactionKernel`, který přímo realizuje rovnice 3.44, 3.45, 3.46 a 3.47.

V rámci kernelu je nejprve provedena kontrola, zda daná částice vůbec vstupuje do interakce porovnáním její objemové hustoty s nastaveným prahem,

viz pseudokód 8. Poté je určena hodnota mřížkové rychlosti a vířivosti na pozici částice interpolací ze sousedních buněk mřížky a spočtena relativní rychlost jako rozdíl rychlosti částice a právě spočítané mřížkové rychlosti na její pozici. Následně jsou spočteny hodnoty $\mathbf{f}_{i,j}^{drag}$, $\mathbf{f}_{i,j}^{lift}$, G_j a A_i viz sekce 3.3.1. Podrobný výpočet je představen na ukázce kódu 5.8.

Pseudokód 8 Pseudokód kernelu `interactionKernel`, který realizuje interakci mezi částicemi a mřížkou. Výpočet sil a množství přeneseného sněhu je podrobněji znázorněn na ukázce kódu 5.8.

```
1:  procedure interactionKernel
2:      idx ← index of particle
3:      if number_density[idx] > interaction_threshold then
4:          x, y, z ← position of idx in grid
5:          interpolated_vel ← velocity on x, y, z
6:          interpolated_vort ← vorticity on x, y, z
7:          relative_vel ← sph_vel[idx] − interpolated_vel
8:          f_drag ← computed drag force
9:          f_lift ← computed lift force
10:         f_total ← f_drag + f_lift
11:         amount_generated ← amount of generated snow smoke
12:         amount_adhered ← amount of adhered snow smoke
13:         amount_total ← amount_generated − amount_adhered
14:         grid density on x, y, z ← amount_total
15:         grid vleocity on x, y, z ← −f_total
16:         sph mass on idx ← −amount_total
17:         sph force on idx ← f_total
```

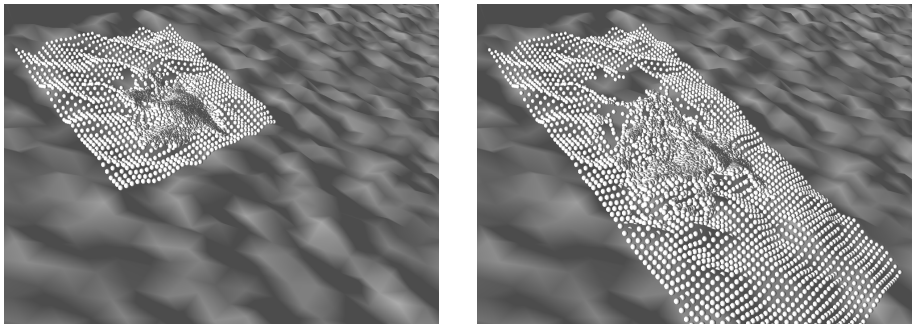
Nastavení interakce se stalo problematickou částí implementace, neboť je založeno na mnoha parametrech. Při dodržení postupů výpočtu a nastavení parametrů dle výchozího článku [5] jsou rychlosti na mřížce prachového sněhu spíše menší a advekční krok se příliš neprojevuje. Tyto rychlosti jsou navíc výrazně ovlivněny změnou velikosti částic a rozměru buněk mřížky, neboť se mění počet částic, které v interakci přispívají do cílové buňky. Snaha zvětšit interakční síly zavedením nového koeficientu však obvykle vedla na opačný extrém, kdy byl prachový sníh v advekčním kroku unášen rychleji, než chování lavin vyžaduje. Nakonec byla z důvodu vyšší stability upřednostněna původní verze.

5.5 Vrstva akumulovaného sněhu

Poměrně překvapivě se problémem při implementaci stalo ztvárnění vrstvy akumulovaného sněhu. Původní řešení navržené v [5] bylo poměrně záhy zavrženo z důvodu plýtvání výpočetními prostředky na velké množství částic v několika vrstvách, které se nikdy nezačnou pohybovat a z důvodu chybějících detailů ve výpočtu prahové hodnoty pro odtržení částice.

V původní metodě je na povrchu terénu v několika vrstvách unifromně

rozmístěno množství částic, které se mohou stát částicemi pohybujícími se v rámci tekoucí vrstvy laviny až po splnění podmínky odtržení. Takové množství částic však je enormní, proto byl návrh zjednodušen pouze na akumulovanou vrstvu tvořenou jedinou úrovní částic (částice akumuované vrstvy jsou umístěny pouze na úrovni terénu, nikoliv nad sebou). Navíc je patrné, že obdobně jako v případě vrstvy prachového sněhu, i vrstva akumulovaného sněhu musí svou velikostí reagovat na změnu velikosti tekoucí vrstvy. Jednoduše řečeno se nesmí stát, že by se částice tekoucí vrstvy vyskytovaly nad terénem, který není pokryt částicemi vrstvy akumulované.



Obrázek 5.6: Zvětšení akumulované vrstvy v závislosti na pohybu tekoucí vrstvy. Zároveň je vidět, že množství částic bylo již lavinou strženo.

Z tohoto důvodu bylo k problému přistoupeno obdobně jako při zvětšování mřížky simulující vrstvu prachového sněhu. Na počátku jsou částice akumulované vrstvy rozmístěny pouze v okolí částic vrstvy tekoucí. Kernel `updateBoxDimensions` uvedený v rámci sekce věnované tekoucí vrstvě tak nyní počítá konvexní obálku všech částic včetně částic akumulované vrstvy. Je zřejmé, že mezi jednotlivými simulačními kroky se rozměr konvexní obálky změní pouze v případě, že se některá z částic tekoucí vrstvy dostala mimo oblast vymezenou částicemi vrstvy akumulované. Tímto způsobem je možno snadno indikovat nutnost zvětšení akumulované vrstvy, což je následně provedeno zvětšením dimenze na dvojnásobek původní velikosti ve směru, ve kterém má dojít ke zvětšení (do kladných nebo záporných hodnot ve směru osy x a obdobně ve směru osy y) a nagenerováním nových částic. O kontrolu a případné zvětšení se stará funkce `resizeAccumulatedLayer` volaná na začátku simulačního kroku (jakmile proběhlo volání `updateBoxDimensions`) v rámci simulace SPH. Pokud má dojít ke zvětšení, je zavolána funkce `placeSnowCover`, která určí oblast, kde mají být nagenerovány nové částice a o jejich rozmístění se postará kernel `placeNewParticles`. Nakonec je opravena velikost konvexní obálky na novou hodnotu.

Nutno zmínit, že korektní generování nových částic je možné pouze tehdy, pokud je zajištěna dostatečná velikost cílových polí, do kterých se hodnoty ukládají. Jedná se o tatáž pole použitá při simulaci tekoucí vrstvy, neboť částice akumulované vrstvy je od tekoucích možno snadno oddělit přidělením číselné hodnoty každému typu (vyžaduje pouze další nové pole). Pokud pole nemají dostatečnou velikost, aby do nich mohly být uloženy hodnoty nových částic, je zavolána funkce `realloc`, která obsah původních polí překopíruje

do zvětšených obdobným způsobem jako při zvětšování mřížky pro prachový sníh (ale není nutné provádět žádnou přeindexaci).

Odtržení částice akumulované vrstvy (a její změna na částici vrstvy tekoucí) je pak také řešeno jednodušším způsobem. Tato úprava vychází z pozorování chování simulace, kdy bylo zjištěno, že rozumných výsledků lze dosáhnout např. i v případě, že na částici akumulované vrstvy působí síla z tekoucí vrstvy přibližně ve směru sklonu terénu. Aby nemohla odtržení způsobit osamocená pohybující se částice, je provedena kontrola také na hustotu v bodě odtržení. K odtržení pak dojde pouze tehdy, je-li dosaženo kritické hustoty.

Jelikož je uvedená metoda podstatným zjednodušením původní metody a zároveň neposkytuje zcela komplexní řešení problému, bylo zvažováno i řešení jiné, inspirované [41]. V tomto případě by terén byl opět pokryt jedinou vrstvou stacionárních částic, které by se však samy nikdy nestaly částicemi tekoucí vrstvy, ale zajišťovaly by pouze interakci. Pokud by na tyto částice působila síla dostatečně velká, aby došlo k odtržení, byla by vygenerována zcela nová částice tekoucí vrstvy v malé vzdálenosti nad interagující částicí (pokud by byla vygenerována přímo v bodě částice, působily by zde příliš silné SPH interakční síly). Nové částice by takto mohly být generovány pouze do té doby, dokud by nebyl vyčerpán sníh, který se při terénu nacházel. Jeho množství by bylo uloženo v nové CUDA textuře pro každý vrchol terénu. Interpolací by bylo určité množství při interakci z nejbližších vrcholů této textury odebráno. Při vykreslování terénu a při kontrole kolizí s terénem by pak byl proces upraven tak, že by se počítalo i s vrstvou sněhu nad původní výškou terénu. Změny v množství sněhu by se tak dynamicky propagovaly přímo do tvaru terénu.

Implementace této metody byla vyzkoušena, již v počátcích se však ukázalo, že výsledek nesplňuje očekávání. Generování nových částic i přes úpravu pozice způsobovalo nestabilitu simulace nebo naopak způsobilo přílišné zhuštění částic bez nárůstu objemu, to vše za výrazného zpomalování celé simulace. Negativně se projevilo i na chování vrstvy prachového sněhu, neboť do interakce s ní vstupovaly příliš velké síly. Na druhou stranu tento pokus přinesl alespoň možnost dalšího případného rozšíření, které by využívalo generování nových částic, neboť s ním implementace již počítá. Lavina by např. svým pohybem mohla strhávat pevné objekty (stromy, kameny), které by se dále rozlamovaly na více kusů. Takové objekty by byly pokryty vrstvou SPH částic, se kterými by interagovaly částice tekoucí vrstvy. V případě rozlomení by bylo nutné nagenerovat nové částice na rovinu zlomu.

5.6 Vizualizace

Jak již bylo zmíněno v kapitole 4, vizualizace částic a hustot v mřížce je řešena jednoduchým způsobem. Na uvedených pozicích však není generován zástupný objekt složený z polygonů, ale samotné pozice jsou poslány do vykreslovacího řetězce jako `GL_POINT_SPRITE`. O vykreslení kruhových objektů správné velikosti se pak starají jen shader programy. Nejprve musí být ve vertex shaderu určena správná velikost bodu v obrazovém souřadnicovém systému.

Toho je dosaženo výpočtem 5.9, kde `distance` určuje vzdálenost kamery od vykreslovaného bodu v souřadnicovém systému kamery.

```
gl_PointSize = (radius * 2.0f * heightOfNearPlane) /
    distance( mvPosition, vec4(0.0, 0.0, 0.0, 1.0));
```

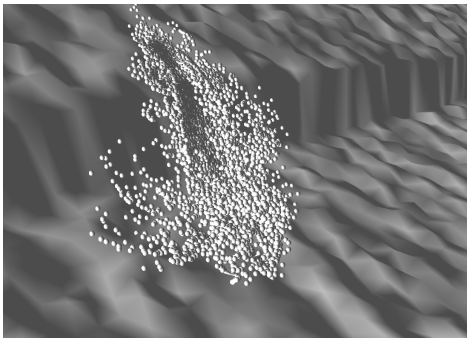
Ukázka kódu 5.9: Výpočet velikosti vykresleného bodu.

Takto získané body jsou ale vykresleny jako čtverce, je proto nutné je dále upravit ve fragment shaderu, kde jsou zahozeny pixely mimo požadovanou kruhovou oblast. Zároveň je spočtena 3D normála, kterou by měla koule o zadaném poloměru na pozici aktuálního pixelu, viz ukázka kódu 5.10. Díky normálovým vektorům je možné aplikovat osvětlení i na `GL_POINT_SPRITE`. Osvětlení se aplikuje pouze u částic SPH, na mřížce jsou naopak hodnoty v buňkách vykresleny s průhledností danou hustotou.

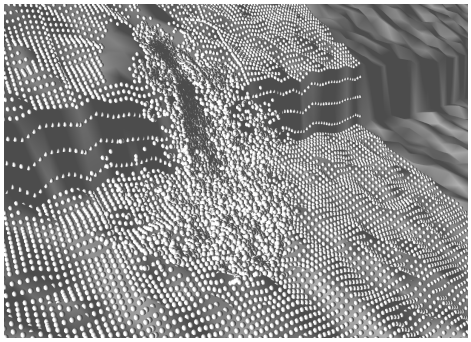
```
N.xy = gl_PointCoord * 2.0 - vec2(1.0);
float mag = dot(N.xy, N.xy);
if (mag > 1.0) discard;
N.z = sqrt(1.0-mag);
```

Ukázka kódu 5.10: Výpočet normály v částicovém fragment shaderu a oříznutí na kruh.

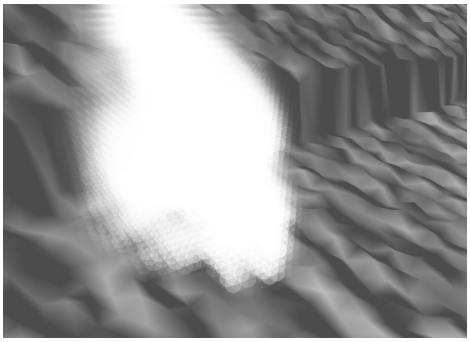
Rychlostní pole mřížky je zobrazeno jednoduchými úsečkami, které zastupují jednotlivé vektory. Aby bylo možné určit směr vektoru, je počáteční bod úsečky obarven modře a koncový červeně.



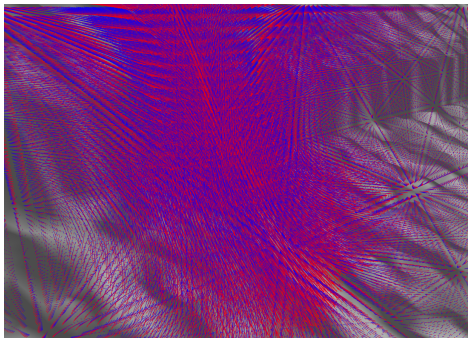
(a): Vizualizace tekoucí vrstvy, částic SPH.



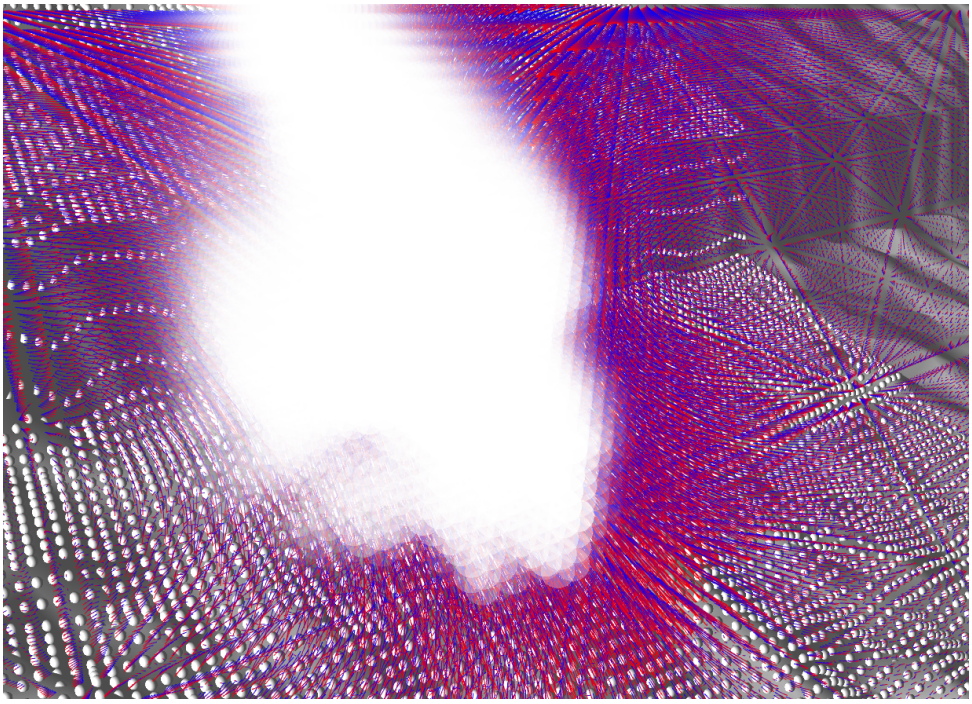
(b): Vizualizace tekoucí a akumulované vrstvy.



(c): Vizualizace vrstvy prachového sněhu, skalárního pole na mřížce.



(d): Vizualizace rychlostního pole na mřížce.



(e): Vizualizace všech částí zároveň.

Obrázek 5.7: Vizualizace různých částí téže scény. Uvedené části je možno zobrazit také zároveň, viz poslední obrázek. Znázorněna je simulace na zvlněné nakloněné rovině.

Kapitola 6
Testování

Vytvářená aplikace byla testována nejen v rámci finálního zhodnocení, ale i v průběhu implementace k ověření funkčnosti a porovnání jednotlivých částí. Implementace i testování probíhaly standardně na notebooku s následujícími parametry:

- MS Windows 7 Home Premium 64-bit
- Intel Core i5 - 3210M @ 2.50GHz
- 6.0 GB RAM
- NVIDIA GeForce GT 630M
- Compute Capability 2.1
- 96 CUDA cores
- 2 streaming multiprocesory
- CUDA Driver version 8.0.44

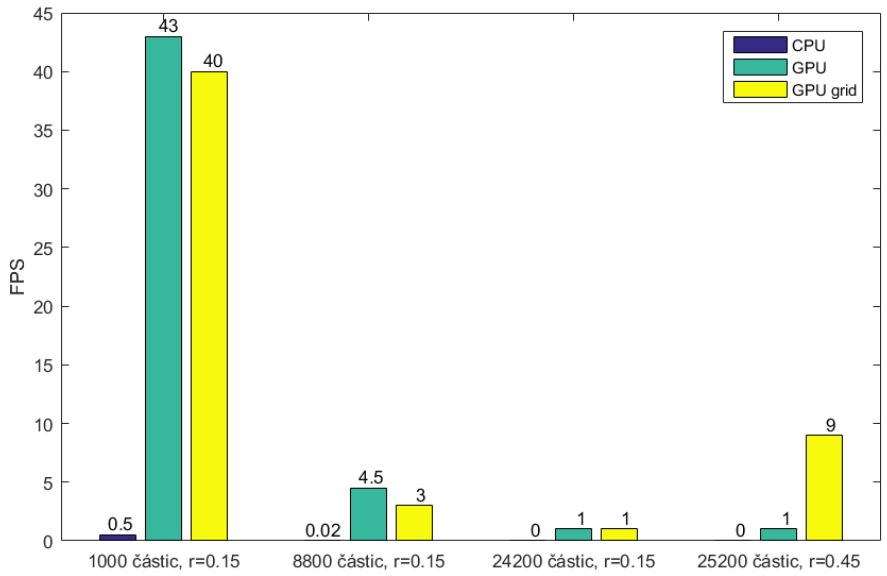
Aplikace byla otestována již ve fázi, kdy simulovala pouze tekoucí vrstvu (SPH). Toto testování bylo zaměřeno na porovnání výkonu implementace na CPU a na GPU. Po dokončení byly vybrány tři odlišné scény, u nichž byl výstup podroben srovnání s videi skutečných lavin. Vizuální výstup simulace byl také porovnán s referenčním řešením z výchozího článku [5].

6.1 SPH

První měření byla provedena už po dokončení samotné vrstvy tekoucího sněhu tvořené systémem SPH. V této fázi bylo provedeno porovnání implementace naivní varianty na CPU (neobsahuje paralelizaci, neobsahuje akcelerační strukturu), se dvěma variantami na GPU (bez akceleřační mřížky a s mřížkou).

Výsledky měření se nachází na obrázku 6.1. Počet snímku za sekundu odpovídá počtu kroků simulace, které je možné danou metodou za sekundu spočítat. Je zřejmé, že využití paralelizace na GPU je pro výpočet simulace pomocí SPH velmi vhodné. Oproti sekvenčnímu řešení na CPU dochází na GPU k mnohonásobnému zrychlení již při použití základního řešení.

Oproti očekávání se však ukázalo, že za daných vstupních podmínek nemusí použití mřížky výpočet dále urychlit, naopak, výpočet může trvat déle než při



Obrázek 6.1: Výsledky měření pro různé konfigurace simulace. Uvedena je průměrná naměřená snímkovací frekvence (FPS). Vždy platí, že $h = 0,8$.

použití jednoduché paralelizace na GPU. Při použití mřížky se sice netestují všechny částice na možného souseda, počet testovaných částic je však stále vysoký (např. v testovací konfiguraci 2, viz obrázek 6.1 může být na souseda testováno i 2500 částic, což je více než čtvrtina). Je to způsobeno malým počtem buněk mřížky. V tomto testovacím scénáři obsahuje mřížka po spuštění simulace $4 \times 4 \times 4$ buněk, při testování 27 buněk na sousednost je tak stále testována rozsáhlá část mřížky. Větší počet buněk však není možno ve scénách s danou konfigurací vytvořit, neboť velikost buňky závisí na parametru h. Snížením hodnoty parametru (a současným zvýšením počtu buněk v mřížce) se ovlivňuje i chování částic; je sice možné takto dosáhnout zrychlení, ale přílišné snížení hodnoty h může vést k totálnímu selhání simulace.

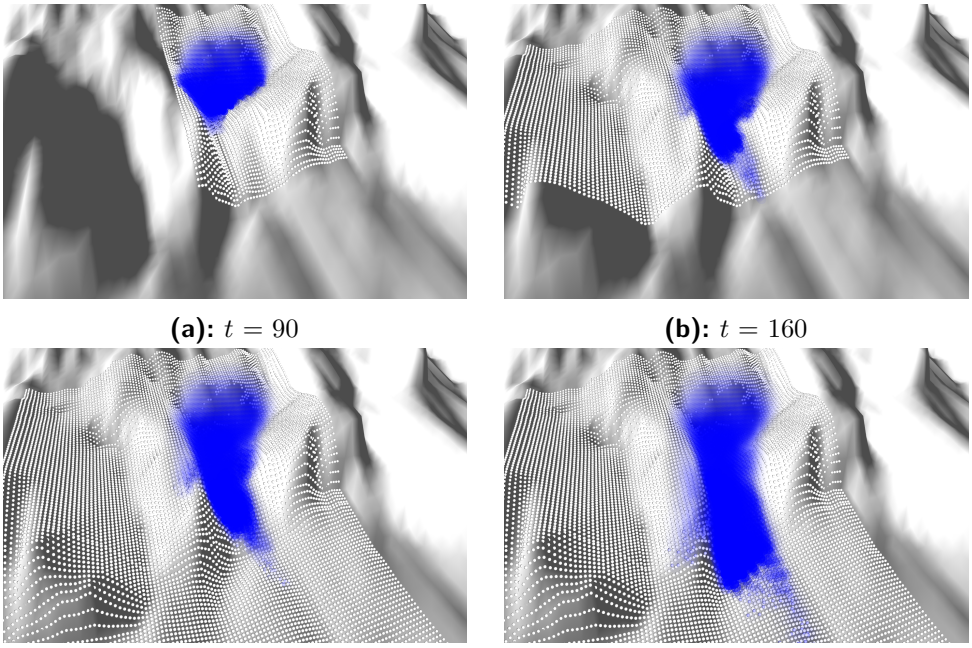
Na druhou stranu je možné rychlost také ovlivnit zvětšením poloměru jednotlivých částic (a s poloměrem související hmotnosti částice). K tomuto účelu byla aplikace otestována i na speciální testovací konfiguraci 4, viz obrázek 6.1. Zvětšením mřížky a současným zvětšením poloměru částice se v jednotlivých buňkách mřížky nachází výrazně méně částic a test sousednosti se provádí méněkrát. Výsledky testu tentokrát očekávání potvrdily. Jediným problémem výrazně větších částic je, že představují příliš velký objem sněhu i v reálném měřítku.

6.2 Testovací scény

Simulace byla otestována na několika scénach a s různým nastavením parametrů. Scény byly zvoleny tak, aby zachycovaly rozdílné chování simulace a aby je bylo možné porovnat s dostupnými videozáznamy skutečných lavin. U každé scény je uveden výstup simulace v různých časových krocích, referenční obrázky a graf naměřené snímkovací frekvence spolu s velikostí mřížky prachového sněhu a počtem částic SPH.

6.2.1 Scéna A

Testovací scéna ukazující chování laviny v úzkém kaňonu. Na výstupu simulace 6.2 je vidět podobné chování jako na referenčním obrázku 6.3, kdy se lavina po opuštění nejužšího místa začíná rozšiřovat. Referenční obrázek byl převzat z videa[1].



(a): $t = 90$ **(b):** $t = 160$

(c): $t = 200$ **(d):** $t = 250$

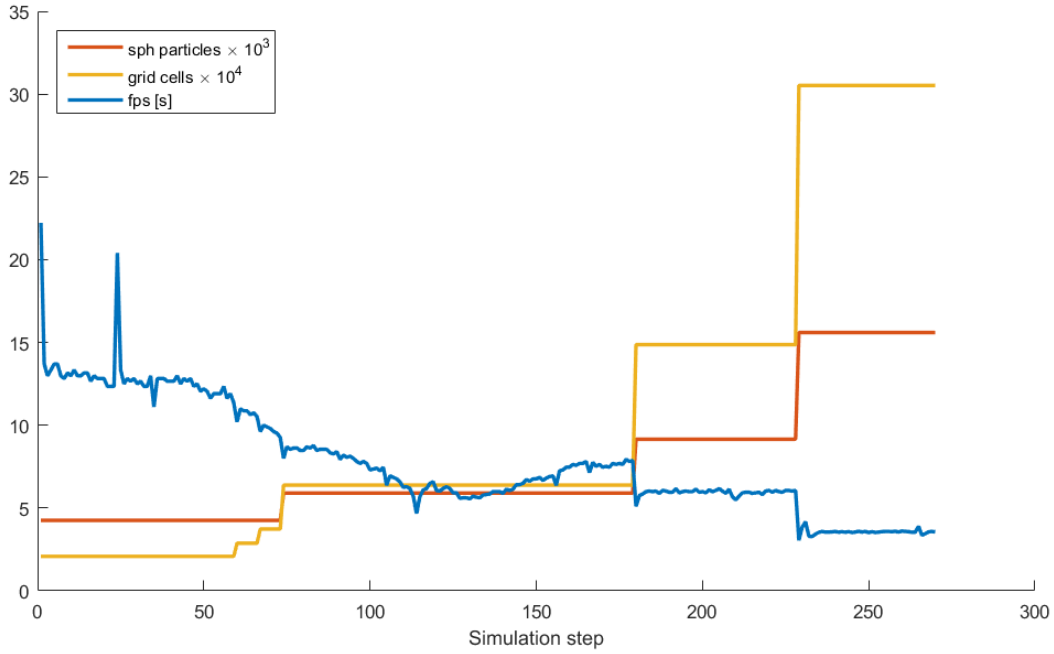
Obrázek 6.2: Výstup aplikace v různých časových krocích pro scénu A. Zvolené parametry: **terén** N46E007.hgt, **počáteční pozice tekoucí vrstvy** $= [409, 847]$, **počáteční rozměr tekoucí vrstvy** $= 2 \times 8$, $\boldsymbol{\rho_0} = 700\,\mathrm{kg/m^3}$, $\boldsymbol{\mu} = 20$, $\boldsymbol{r} = 0.16\,\mathrm{m}$, $\boldsymbol{h} = 0.45$, **grid step** $= 0.5$, $\boldsymbol{k_{drag}} = 0.81$, $\boldsymbol{k_{lift}} = 0.2$, $\boldsymbol{G_j} = 0.1$, $\boldsymbol{A_i} = 0.2$, $\boldsymbol{\epsilon} = 0.5$, $\boldsymbol{diff} = 10$, **interaction threshold** $= 200$.

6.2.2 Scéna B

Testovací scéna ukazující chování laviny na pozvolném svahu bez výrazných nerovností. Tentokrát byla zvolena menší velikost částic a jemnější rozlišení

[1] `https://www.youtube.com/watch?v=JUhmLs5qdzA`

Obrázek 6.3: Lavina v oblasti Bernských Alp.



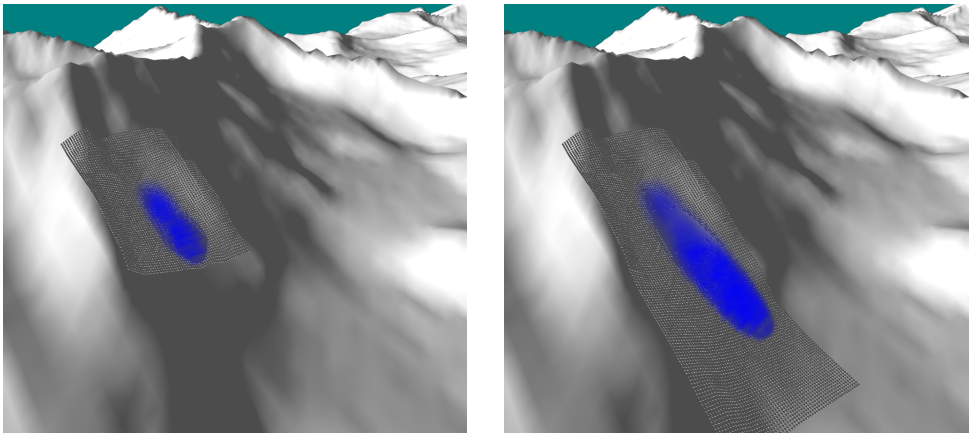
Obrázek 6.4: Výsledky naměřené pro scénu A. Zobrazen je počet částic (červeně), počet buněk mřížky vrstvy prachového sněhu (žlutě) a snímkovací frekvence, které bylo dosaženo (modře).

mřížky prachového sněhu. Oproti předcházející scéně je na obrázku 6.5 vidět, že na přímém svahu dochází k menšímu rozvoji oblaku prachového sněhu, což se shoduje s referenčním ukázkou 6.6. Referenční obrázek byl převzat z videa[2].

6.2.3 Scéna C

Porovnání s uměle vyvolanými lavinami ve švýcarském kantonu Valais, poblíž obce Anzère. Oproti předcházejícím scénám byl načtený terén upraven náhodným zvlněním. Podobně jako v reálném světě, viz 6.9, i simulovaná lavina 6.8 se dělí do několika proudů a je generován výrazný oblak prachového sněhu.

[2] `https://www.youtube.com/watch?v=xxm4VeUpo9c`



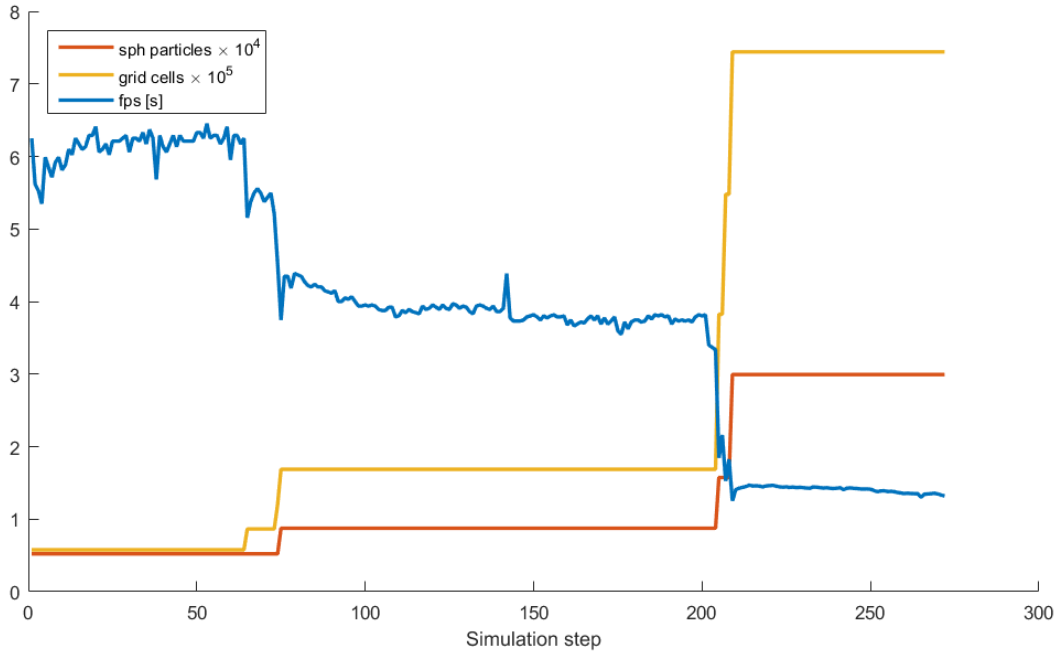
(a): $t = 90$ **(b):** $t = 160$

Obrázek 6.5: Výstup aplikace v různých časových krocích pro scénu B. Zvolené parametry: **terén** N49E020.hgt, **počáteční pozice tekoucí vrstvy** $= [124, 583]$, **počáteční rozměr tekoucí vrstvy** $= 2 \times 1$, $\boldsymbol{\rho_0} = 500\,\mathrm{kg/m^3}$, $\boldsymbol{\mu} = 25$, $\boldsymbol{r} = 0.08\,\mathrm{m}$, $\boldsymbol{h} = 0.5$, **grid step** $= 0.25$, $\boldsymbol{k_{drag}} = 1.0$, $\boldsymbol{k_{lift}} = 0.45$, $\boldsymbol{G_j} = 0.35$, $\boldsymbol{A_i} = 0.2$, $\boldsymbol{\epsilon} = 0.5$, $\boldsymbol{diff} = 100$, **interaction threshold** $= 1000$.

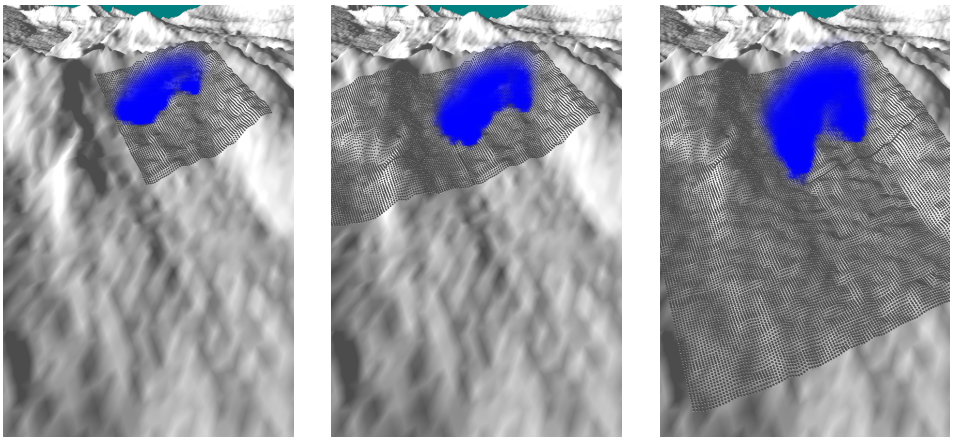
Obrázek 6.6: Lavina v Mlynické dolině, Vysoké Tatry.

Referenční obrázek byl převzat z videa[3].

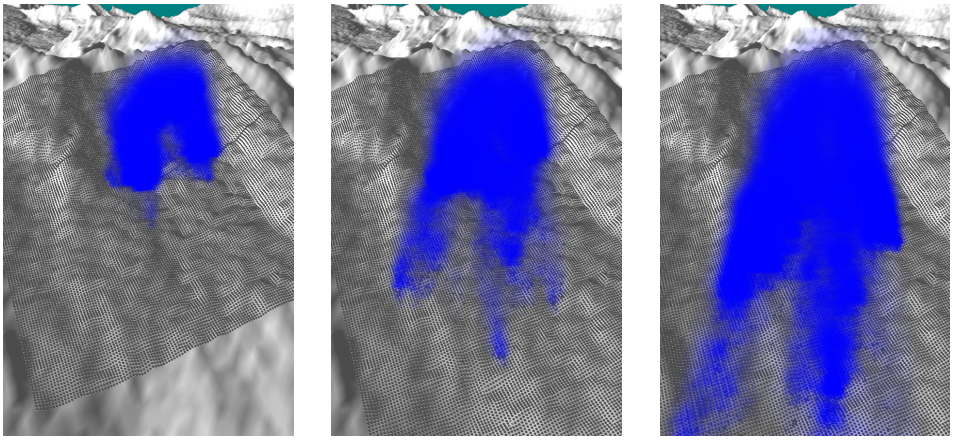
[3] `https://www.youtube.com/watch?v=LVnTp0toZL0`



Obrázek 6.7: Výsledky naměřené pro scénu B. Počet částic a množství buněk mřížky tentokát v o řád větším měřítku.



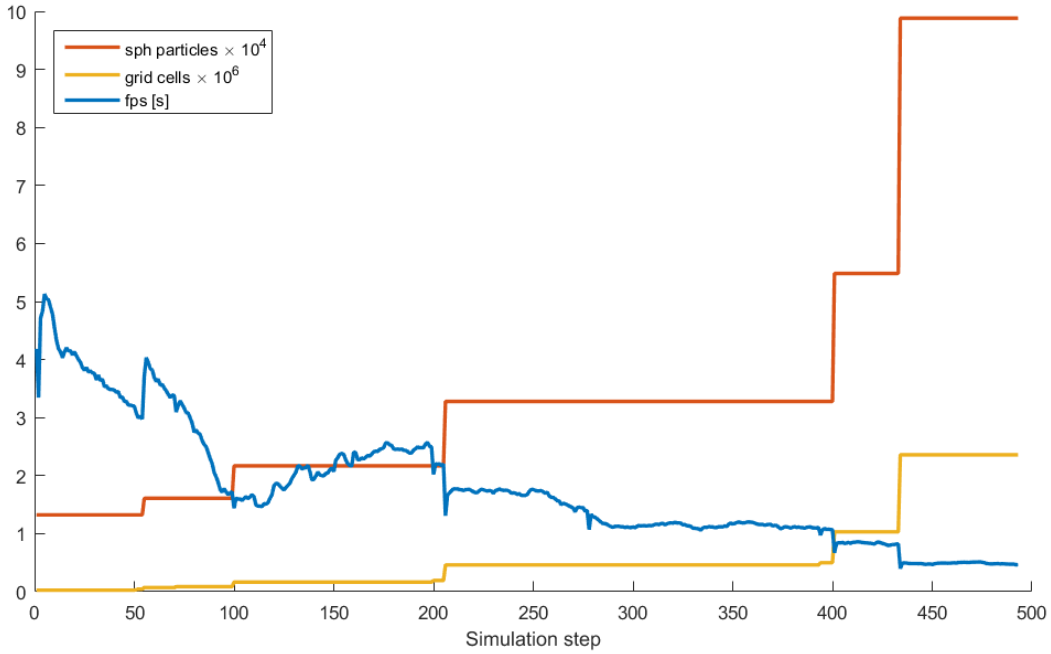
(a): $t = 80$ **(b):** $t = 120$ **(c):** $t = 180$



(d): $t = 240$ **(e):** $t = 350$ **(f):** $t = 450$

Obrázek 6.8: Výstup aplikace v různých časových krocích pro scénu C. Zvolené parametry: **terén** N46E007.hgt, **počáteční pozice tekoucí vrstvy** $= [990, 990]$, **počáteční rozměr tekoucí vrstvy** $= 4 \times 10$, $\boldsymbol{\rho_0} = 550\,\mathrm{kg/m^3}$, $\boldsymbol{\mu} = 60$, $\boldsymbol{r} = 0.14\,\mathrm{m}$, $\boldsymbol{h} = 0.43$, **grid step** $= 0.5$, $\boldsymbol{k_{drag}} = 0.85$, $\boldsymbol{k_{lift}} = 0.35$, $\boldsymbol{G_j} = 0.25$, $\boldsymbol{A_i} = 0.2$, $\boldsymbol{\epsilon} = 0.7$, $\boldsymbol{diff} = 25$, **interaction threshold** $= 750$, **náhodné zvlnění terénu** $= 0.5$.

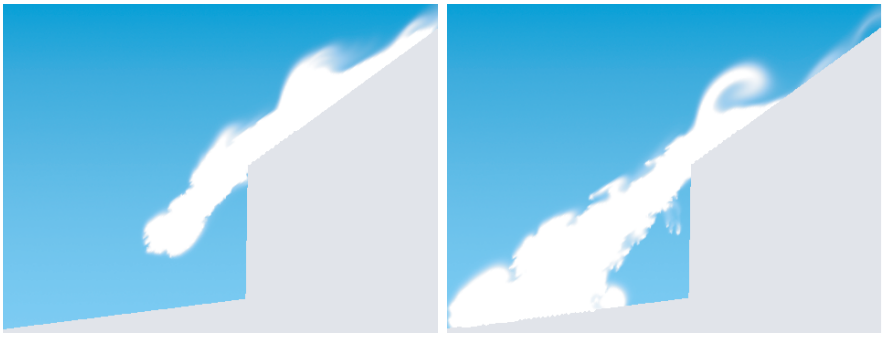
Obrázek 6.9: Uměle vyvolané laviny poblíž švýcarského Anzère.



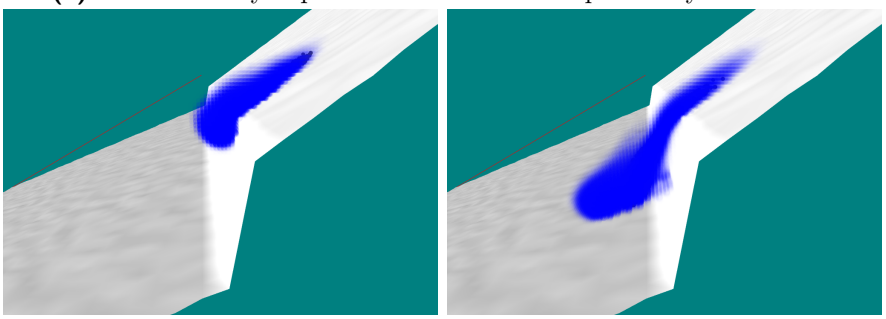
Obrázek 6.10: Výsledky naměřené pro scénu C.

6.3 Porovnání s referenčním řešením

Práce [5] poskytuje pouze dva 3D obrázky výstupu implementované metody, viz obrázek 3.12. Bylo by však značně obtížné nalézt natolik podobný terén, aby mohl být výstup implementované aplikace porovnán. Proto bylo přistoupeno k porovnání s 2D výstupem referenční metody, které je uvedeno na obrázku 6.11.



(a): Referenční výstup na nakloněné rovině s proměnným sklonem.



(b): Výsledek simulace v podobné scéně jako v referenční ukázce.

Obrázek 6.11: Porovnání výsledků s referenční metodou předloženou v [5]. Výsledky jsou do značné míry ovlivněny rozdílnou dimenzionalitou simulace, kdy referenční řešení probíhá pouze ve 2D a navíc je použita odlišná vykreslovací metoda. Vrstva prachového sněhu ve výsledné ukázce byla zobrazena v nereálných barvách pro zvýšení kontrastu.

6.4 Zhodnocení

Implementovaná metoda kombinující mřížkovou metodu a SPH poskytuje uspokojivé vizuální výsledky pro simulaci lavin kombinovaného typu. Během testování bylo dosaženo výpočetní rychlosti v řádu jednotek snímků za sekundu, což je pro tak komplexní metodu a několik let starý notebook, na kterém testování probíhalo, přijatelná hodnota. Je nutné si však uvědomit, že takových časů by nebylo možné dosáhnout bez využití paralelizace na GPU. Jak SPH, tak mřížková metoda, jsou výpočetně náročné metody samy o sobě a jejich kombinací se ještě zvýrazní jejich nároky. V případě SPH roste s počtem částic i počet testů sousednosti, pokud není využito

akcelerační struktury, je tato závislost kvadratická. U mřížkové metody se výrazné zpomalení při využití GPU paralelismu projevuje až okolo 10^6 buněk, v takovém případě je navíc potřeba minimálně 52 MB paměti pro uložení 13 potřebných polí mřížkových hodnot. Při parametrech dnešních GPU se velikost spotřebované paměti nejeví jako zásadní problém, velké množství přístupů do paměti však zůstává a projevuje se na prodloužení výpočetních časů.

Pokud by aplikace vznikající v rámci této práce nevyužívala technologie CUDA, byly by časy potřebné ke zpracování jednoho kroku simulace výrazně delší a testování i použitelnost aplikace by byly ztíženy. Pokud se podíváme na výsledky dosažené ve výchozím článku [5], zjistíme, že při 3D simulaci na stroji s Intel Core 2 DUO 3.33 GHz CPU, při počtu SPH částic 10,000 – 22,000 v tekoucí vrstvě a mřížce o 204,000 buňkách bylo k výpočtu jediného simulačního kroku potřeba v průměru 8 sekund. Zde překvapí poměrně malý počet buněk mřížky, jejíž rozměr je navíc v průběhu simulace pevně nastaven. Tyto výsledky se díky použití GPU paralelismu podařilo překonat, mřížka navíc dokáže měnit svůj rozměr v závislosti na průběhu simulace. Zvětšení mřížky pak sice znamená jisté zpomalení výpočtu v daném simulačním kroku, ale je natolik malé, že bohatě vynahrazuje zpomalení, které by způsobilo využití enormně velké mřížky bez zvětšování po celou dobu simulace.

Kapitola 7

Závěr

Cílem práce bylo na základě prostudované literatury o tvorbě lavin a existujících řešeních jejich simulace navrhnout a implementovat aplikaci, která by umožnila simulovat laviny kombinovaného typu v terénu popsaném výškovou mapou.

Úspěšně byla navržena a implementována aplikace využívající kombinaci Lagrangeovského systému (částice SPH) pro tekoucí vrstvu laviny tvořenou sněhem o vyšší hustotě a systému Eulerovského (mřížková metoda) pro oblak prachového sněhu. Aplikace také zjednodušeně simuluje strhávání sněhu akumulovaného při povrchu terénu pohybující se lavinou. Úspěšné dosažení cílů bylo podpořeno využitím paralelizace na GPU v technologii CUDA, díky čemuž může simulace probíhat v rozumném čase.

Jednu ze základních překážek při návrhu představovala předem neznámá prostorová náročnost pohybující se laviny. Bylo nutné nalézt řešení bez zbytečného plýtvání výpočetními prostředky, které by ale zároveň umožnilo kontinuální průběh simulace v libovolném terénu bez dalších požadavků na vstup od uživatele. Tuto překážku se podařilo úspěšně překonat využitím automaticky se zvětšujících struktur, což při využití technologie CUDA představovalo výraznou část implementace.

Implementovaná metoda poskytuje uspokojivé vizuální výsledky, viz kapitola 6. Jelikož pokročilá vizualizace nebyla cílem této práce, využívá výsledná aplikace pouze jednoduché vizualizační metody, čímž je výstup do značné míry ovlivněn. Do budoucna se tedy nabízí použití pokročilejších zobrazovacích technik jak pro tekoucí vrstvu, tak pro prachový sníh.

Zároveň se nabízí přidání rozšíření v podobě pevných komplexnějších objektů (stromy, kameny), které by mohly být lavinou strženy. Toto rozšíření by vyžadovalo pokrytí objektů vrstvou SPH částic, které by vstupovaly do interakce s tekoucí vrstvou. Bylo by zároveň nutné implementovat odlišný strhávací mechanismus.

V neposlední řadě by bylo vhodné do simulace zahrnout i fyzikálně založené počáteční odtržení laviny, neboť nyní je tekoucí vrstva pouze umístěna na terén a uvolněna. Takové uvolnění neodpovídá realitě, kdy se části sněhové masy odlamují postupně podél zlomové linie.

Příloha A

Literatura

[1] Avalanche Skills Training - OVERhang - Outdoor Vertical Education and Recreation - Specialists in the Instruction of Outdoor, Wilderness, and Occupational Safety, as Well as an Indoor Climbing Gym - Prince George, BC: *Falling avalanche*. [online, image], www: `http://overhang.ca/training/courses/avalanche-skills-training`, cit. 23. 4. 2017.

[2] Christophe Ancey: *Snow Avalanches*. Cemagref, unité Erosion Torrentielle, Neige et Avalanches, Domaine Universitaire, 38402 Saint-Martin-d'Hères Cedex, France.

[3] *EAWS - European Avalanche Warning Services Glossary*. [online], www: `http://www.avalanches.org/eaws/en/includes/glossary/glossary_en_all.html`, cit. 20. 4. 2017.

[4] Leif Kåre Hornnes Yndestad: *Particle-based Powder-snow Avalanche Simulation Using GPU*. Theses, Norwegian University of Science and Technology, Department of Computer and Information Science, 2011.

[5] Yusuke Tsuda, Yonghao Yue, Yoshinori Dobashi, Tomoyuki Nishita: *Visual Simulation of Mixed-motion Avalanches with Interactions Between Snow Layers*. The Visual Computer, Vol.26, Issue 6-8, strany 883-891, Springer-Verlag, 2010.

[6] Jeff Levison: *Avalanches: Physics in Motion*. Physics 212 project, University of Alaska Fairbanks, 2004, www: `http://ffden-2.phys.uaf.edu/212_fall2009.web/Jeff_Levison/Jeff%20Levison%201%20Home.html`, cit. 15. 1. 2017.

[7] Martin Bláha: *Simulace akumulace sněhu*. Diplomová práce, České vysoké učení technické v Praze, Fakulta elektrotechnická, 2015.

[8] Jeff Levison: *The Physics of a Snow Avalanche*. Physics 211x F05, 2004, www: `http://ffden-2.phys.uaf.edu/211_fall2004.web.dir/Jeff_Levison/The%20Physics%20of%20a%20Snow%20Avalanche.htm`, cit. 17. 1. 2017.

[9] David Eyre: *A Mathematical Look at Snow and Avalanches*. Forest Service Utah Avalanche Forecast Center Avalanche Advisory, November 22, 1999,

www: `http://www.math.utah.edu/~eyre/lectures/snow/`, cit. 17. 1. 2017.

[10] Jürg Schweizer, J. Bruce Jamieson, Martin Schneebeli: *Snow avalanche formation*. Reviews of Geophysics, 41(4), DOI: 10.1029/2002RG000123. ISSN 8755-1209, 2003.

[11] Xiaoyi HeLi-Shi Luo: *Lattice Boltzmann Model for the Incompressible Navier–Stokes Equation*. LS. Journal of Statistical Physics (1997) 88: 927. doi:10.1023/B:JOSS.0000015179.12689.e4.

[12] Ryo Miyazaki, Satoru Yoshida, Yoshinori Dobashi, Tomoyuki Nishita: *A Method for Modeling Clouds based on Atmospheric Fluid Dynamics*. Proceeding PG '01 Proceedings of the 9th Pacific Conference on Computer Graphics and Applications, ISBN:0-7695-1227-5, strana 363, 2001.

[13] Robert Bridson: *Fluid Simulation for Computer Graphics, Second Edition*. ISBN: 1482232839, 9781482232837, 276 stran, Taylor & Francis, 2015.

[14] Jos Stam: *Stable fluids*. SIGGRAPH '99 Proceedings of the 26th annual conference on Computer graphics and interactive techniques, strany 121-128.

[15] Gonçalo Amador, Abel Gomes: *A CUDA-Based Implementation of Stable Fluids in 3D with Internal and Moving Boundaries*. Proceeding ICCSA '10 Proceedings of the 2010 International Conference on Computational Science and Its Applications, strany 118-128, 2010.

[16] Denizhan Güçer, Halil Bülent: *Simulation of a flowing snow avalanche using molecular dynamics*. Turkish Journal of Electrical Engineering & Computer Sciences. Vol.22, No.6, strany 1596-1610, 2014.

[17] Thorsten Pöschel, Volkhard Buchholtz: *Molecular dynamics of arbitrarily shaped granular particles*. Journal of Physics I France 5, 1431.1455, 1995.

[18] Matthias Müller, David Charypar, Markus Gross: *Particle-Based Fluid Simulation for Interactive Applications*. Eurographics/SIGGRAPH Symposium on Computer Animation, 2003.

[19] Rade Vignjevic: *Review of Development of the Smooth Particle Hydrodynamics (SPH) Method*. Crashworthiness, Impact and Structural Mechanics (CISM) Cranfield University, UK, 2009.

[20] Markus Becker, Matthias Teschner: *Weakly compressible SPH for free surface flows*. Eurographics/SIGGRAPH Symposium on Computer Animation, 2007.

[21] Ondřej Novák: *Simulace viskózních kapalin*. Diplomová práce, České vysoké učení technické v Praze, Fakulta elektrotechnická, 2007.

[22] Juraj Onderik, Roman Ďurikovič: *Efficient Neighbor Search for Particle-based Fluids*. Journal of the Applied Mathematics, Statistics and Informatics (JAMSI), 2 (2007), No. 3.

[23] Chris Priscott: *3D Langrangian Fluid Solver using SPH approximations*. MSc CAVE 09-10, August 17, 2010.

[24] Matthias Teschner, Bruno Heidelberger, Matthias Müller, Danat Pomeranets, Markus Gross: *Optimized Spatial Hashing for Collision Detection of Deformable Objects*. Proceedings of Vision, Modeling, Visualization VMV'03, strany 47-54, 2003.

[25] Øystein E. Krog, Anne C. Elster: *Fast GPU-based Fluid Simulations Using SPH*. Dept. of Computer and Information Science Norwegian University of Science and Technology (NTNU), 2012.

[26] Afonso Paiva, Fabiano Petronetto, Thomas Lewiner, Geovan Tavares: *Particle-based viscoplastic fluid/solid simulation*. Computer-Aided Design 41(4), strany 306-314, April 2009.

[27] Eduardo W. V. Chaves: *Notes on Continuum Mechanics*, kapitola 1 Tensors. ISBN 978-94-007-5986-2, ISSN 1877-7341, Springer, 2013.

[28] Jos Stam: *Real-Time Fluid Dynamics for Games*. Proc. of the Game Developer Conference, 2003.

[29] Tetsuya Takahashi, Issei Fujishiro, Tomoyuki Nishita: *Visual Simulation of Compressible Snow with Friction and Cohesion*. NICOGRAPH International, strany 35-42, 2014.

[30] Yongning Zhu, Robert Bridson: *Animating Sand as a Fluid*. ACM Transactions on Graphics, Vol. 24, No. 3, strany 965–972, 2005.

[31] Ronald Fedkiw, Jos Stam, Henrik Wann Jensen: *Visual Simulation of Smoke*. SIGGRAPH 2001 Proceedings, 1-8.

[32] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, Andrew Selle: *A material point method for snow simulation*. Visual Simulation of Smoke, ACM Transactions on Graphics (TOG), v.32 n.4, July 2013.

[33] David Jevans, Brian Wyvill: *Ray Tracing Implicit Surfaces*. Department of Computer Science, University of Calgary.

[34] Pavel Tišnovský: *Implicitní plochy (metaballs) v POV-Rayi*. Článek seriálu Vykreslujeme 3D scény s POV-Ray [online], 8. 4. 2008, www: `https://www.root.cz/clanky/implicitni-plochy-metaballs-v-pov-rayi/`, cit. 9. 4. 2017.

[35] Tomoyuki Nishita, Hiroshi Iwasaki, Yoshinori Dobashi, Eihachiro Nakamae: *A Modeling and Rendering Method for Snow by Using Metaballs*.

EUROGRAPHICS '97 / D. Fellner and L. Szirmay-Kalos, Volume 16, (1997), Number 3.

[36] Keenan Crane, Ignacio Llamas, Sarah Tariq: *GPU Gems 3, Chapter 30. Real-Time Simulation and Rendering of 3D Fluids.*

[37] Ondrej Kallo: *Simulácia a vizualizácia turbulencií prúdu vzduchu v okolí prekážok*, diplomová práce, Slovenská technická univerzita v Bratislave, Fakulta informatiky a informačných technológií, květen 2011.

[38] Jonathan de Ferranti, Christoph Hormann: *Digital elevation data*, [online], www:`http://viewfinderpanoramas.org/dem3.html`, cit. 29. 4. 2017.

[39] David Vyvlečka: *Grafické uživatelské rozhraní pro aplikaci Admesh*, České vysoké učení technické v Praze, Fakulta informačních technologií, bakalářská práce, 2015.

[40] opengl-tutorial: *Picking with an OpenGL hack*, [online], www:`http://www.opengl-tutorial.org/miscellaneous/clicking-on-objects/picking-with-an-opengl-hack/`, cit. 29. 4. 2017.

[41] Petr Kryštof, Bedřich Beneš, Jaroslav Křivánek, Ondřej Šťáva: *Hydraulic Erosion Using Smoothed Particle Hydrodynamics*, Computer Graphics Forum 28(2), strany 219 - 228, duben 2009.

Příloha B

Uživatelská příručka

B.1 Instalace

Jelikož implementovaná aplikace využívá technologie CUDA, která je závislá na překladu pro specifický hardware, není možné s prací dodat zároveň binární verzi aplikace připravenou ke spuštění. Kompilaci zdrojových souborů je nutné provést pomocí standardního CMake (`https://cmake.org/`). Zároveň je nutné splnit další podmínky:

- Kompilace probíhá na zařízení s podporou CUDA s compute capability (CC) alespoň verze 2.0.
- Je nainstalována knihovna Qt, alespoň verze 5.4 (`https://www.qt.io/`).
- Je nainstalována knihovna OpenGL verze alespoň 4.1 (vše potřebné je zajištěno instalací knihovny Qt).
- Zařízení podporuje shader programy a GLSL verze alespoň 4.10.6.

B.2 Ovládání

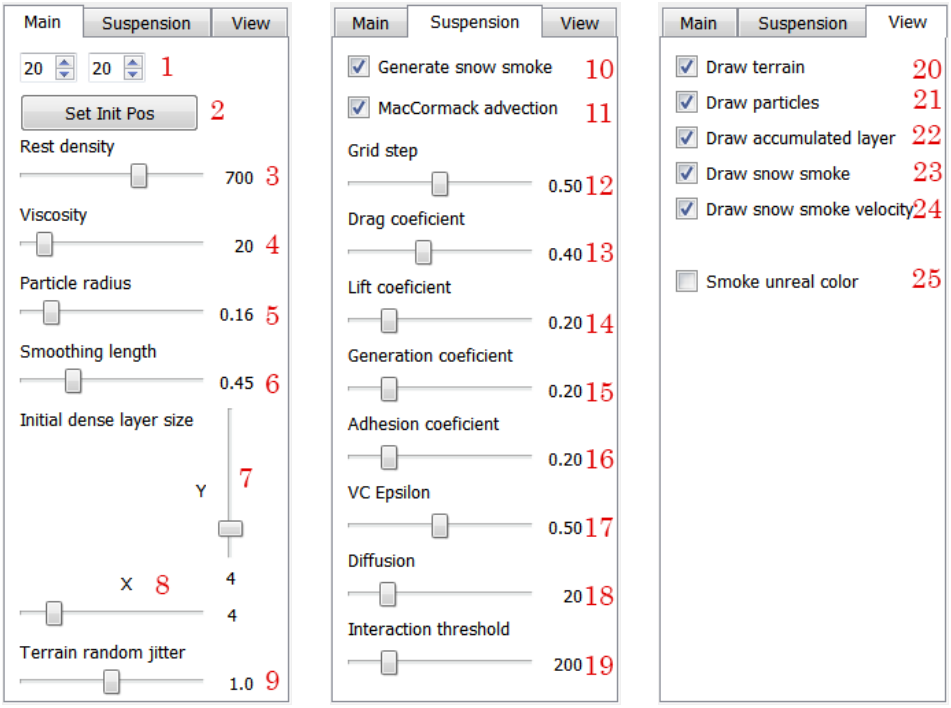
Po spuštění se aplikace nachází v připraveném režimu, kdy je vygenerován implicitní terén a tekoucí vrstva umístěna na výchozí pozici. Odlišný terén je možné načíst přes horní menu. Po načtení terénu umístěte tekoucí vrstvu na pozici buď dvojklikem na terén nebo přes ovládací panel, viz dále.

B.2.1 Hlavní okno simulace

- **Levé tlačítko myši** – drag pro rotaci, dvojklik na terén pro výběr iniciální pozice laviny
- **Pravé tlačítko myši** – drag pro translaci
- **Kolečko myši** – přiblížení nebo oddálení scény

B.2.2 Ovládací panel simulace

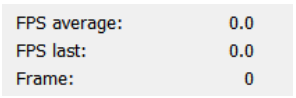
Popis všech částí ovládacího panelu je uveden na obrázku B.1. Popsány jsou položky označené červenými čísly. Některé položky (např. velikost buňky mřížky) není možné měnit v průběhu simulace, nastavená hodnota se projeví až po resetování. Pokud uživatel tyto hodnoty změní, je na nutnost resetovat simulaci upozorněn textovou formou pod panelem nastavení.



Obrázek B.1: Popis ovládacího panelu.

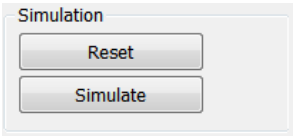
Popis jednotlivých položek: **1** – nastavení počátečních souřadnic částic tekoucí vrstvy, pokud není využito kliknutí přímo na terén; bude aplikováno po stisknutí **2**. **3** – nastavení klidové hustoty. **4** – nastavení viskozity. **5** – poloměr částice. **6** – parametr h (pro jádrové funkce a velikost akcelerační mřížky). **7** – počáteční velikost tekoucí vrstvy ve směru osy x. **8** – počáteční velikost tekoucí vrstvy ve směru osy y. **9** – velikost náhodně generovaného zvlnění terénu. **10** – vypnutí / zapnutí simulace prachového sněhu. **11** – vypnutí / zapnutí Mac Cormackova avdekčního schématu. **12** – délka hrany buňky mřížky prachového sněhu (ovlivňuje detail a rychlost simulace). **13** – velikost parametru k_{drag}. **14** – velikost parametru k_{lift}. **15** – velikost parametru generování prachového sněhu C_g. **16** – velikost parametru adheze prachového sněhu C_a. **17** – velikost parametru pro víry generované metodou vorticity confinement. **18** – míra difuze prachového sněhu. **19** – prahová hodnota početní hustoty částic pro interakci, s větší hodnotou vstupuje do interakce více částic. **20** – zobrazení terénu. **21** – zobrazení částic SPH. **22** – zobrazení akumulované vrstvy. **23** – zobrazení vrstvy prachového sněhu. **24** – zobrazení rychlostního pole na mřížce prachového sněhu. **25** – zobrazení prachového

sněhu v nereálné barvě pro vyšší kontrast.



Obrázek B.2: Statistiky simulace zobrazené v rámci aplikace. Je vidět průměrná snímkovací frekvence, snímkovací frekvence v posledním simulačním kroku a pořadí posledního zpracovaného snímku.

Základní ovládání simulace je uvedeno na obrázku B.3. Tlačítko `Simulate` spustí nebo pozastaví simulaci. Při pozastavení zůstane simulace v poslední konfiguraci, v jaké se nacházela a nedochází k resetování žádných hodnot. Tlačítko `Reset` pak slouží právě k resetování simulace, nastavení původní pozice tekoucí vrstvy a zároveň k aplikaci všech nastavení, která uživatel učinil během simulace (ale nebyla aplikována). Toto tlačítko je možné použít pouze pokud není simulace spuštěna.



Obrázek B.3: Základní ovládání simulace.