VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SIMULACE TEKUTIN A PLYNŮ

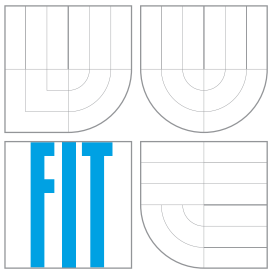
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAKUB ŠTAMBACHR

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SIMULACE TEKUTIN A PLYNŮ

FLUID SIMULATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAKUB ŠTAMBACHR

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. RADOVAN JOŠTH

BRNO 2011

Abstrakt

Tato diplomová práce se zabývá simulacemi kapalin a plynů, obzvláště pak počítačovou simulací toku viskózních newtonských kapalin s volným povrchem. Hlavním cílem této práce je implementovat výkonný simulační model, využívající paralelní architektury grafických karet k obecným výpočtům. K implementaci jsem zvolil Smoothed Particle Hydrodynamics, lagrangeovskou metodu založenou na částicích. Podstatnou část této práce tvoří analýza rychlosti implementovaného algoritmu, srovnání dosažených výsledků s pracemi jiných autorů a demonstrace přínosu použití grafických karet oproti klasické implementaci pro CPU. Výstupem práce je interaktivní program umožňující simulovat (a vizualizovat) vodě podobné kapaliny v reálném čase.

Abstract

This diploma thesis addresses the problem of liquid and gas simulation, it particularly deals with computer simulation of flow of viscous newtonian liquids with a free surface. A main goal of this work is to create an efficient simulation model, utilizing the benefits of current GPU parallel architecture for general-purpose computing. I chose to implement Smoothed Particle Hydrodynamics, a lagrangian particle-based method. A significant portion of this thesis consists of speed analysis of the implemented algorithm, comparison with other authors' achievements in the field and a demonstration of benefits brought by GPU involvement in the computation. As an output of the thesis I present an interactive computer program that allows for real-time simulation (and visualization) of water-like fluids.

Klíčová slova

Simulace, kapaliny, plyny, částicový systém, SPH, CUDA

Keywords

Computer simulation, liquid, gas, particle system, SPH, CUDA

Citace

Jakub Štambachr: Simulace tekutin a plynů, diplomová práce, Brno, FIT VUT v Brne, 2011

Simulace tekutin a plynů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Radovana Joštha

.....
Jakub Štambachr
24. května 2011

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

S kapalinami se setkáváme v každodenním životě téměř na každém kroku. Intuitivně známe a dovedeme si představit jejich chování. O to je možná překvapivější složitost modelování a simulace kapalin. V dnešní době neexistuje univerzální model postihující všechny vlastnosti reálných kapalin a některé jevy související s dynamikou kapalin, jako je turbulence, neumíme s uspokojivou přesností a zároveň v přijatelném čase simulovat vůbec.

Vědní disciplína zabývající se simulováním chování kapalin se nazývá výpočetní dynamika tekutin (Computational Fluid Dynamics). Výzkum prováděný v této oblasti má význam jak v aplikacích inženýrských, kde se nehledí tolik na rychlost simulace, jako na vysokou přesnost, ale také — a to stále častěji — v počítačové grafice. Simulace kapalin v počítačové grafice (herní, filmový průmysl...) mají oproti inženýrským simulacím v podstatě opačné priority. Přesnost simulace ustupuje rychlosti výpočtu až na mez, při které simulace probíhá v reálném čase. Simulovat kapaliny v reálném čase, je ve větším měřítku stále velmi obtížný úkol, a to i bez větších požadavků na přesnost.

Většina fyzikálních modelů kapalin vyplývá z Navier-Stokesových rovnic. K jejich řešení existují dva základní přístupy. Jedná se o přístup eulerovský, ve kterém je vývoj simulované tekutiny v čase sledován na prostorové struktuře a o lagrangeovský přístup, kdy je tekutina představována množinou částic, kde každá částice nese vlastnosti tekutiny s sebou. Klasickým případem částicové metody pro simulaci tekutin je metoda Smoothed Particle Hydrodynamics.

Jak jsem již zmínil výše, dosažení simulace v reálném čase je při větším měřítku modelu velmi obtížné. Proto jsme se mnohem častěji mohli setkat se simulátory pracujícími offline. Interaktivní simulace mají ovšem mnoho zajímavých výhod. Interaktivita urychluje vývoj jak matematických modelů, tak implementací. Urychluje proces učení, kdy při změně parametrů modelu můžeme okamžitě pozorovat výsledky.

S nástupem nových generací grafických karet, které jsou stále častěji využívány k fyzikálním výpočtům, se situace rapidně mění a snaha o vývoj interaktivních simulátorů je stále častější. Nespornou výhodou grafických karet pro simulaci tekutin je jejich paralelní architektura, která pro vhodné algoritmy zajišťuje obrovský nárůst výpočetního výkonu. Průkopníkem na tomto poli je firma NVIDIA a její technologie CUDA, která umožňuje poměrně snadnou implementaci paralelních výpočtů na grafických procesorech. Další nespornou výhodou je možnost technologie CUDA spolupracovat přímo s knihovnami jako OpenGL a DirectX, čímž odpadá nutnost přenášet data mezi hlavní pamětí počítače a pamětí grafické karty.

Je třeba poznamenat, že v posledním desetiletí je vývoj grafických procesorů a specializovaného hardware poháněn zejména herním a zábavním průmyslem. S tím souvisí

stále větší požadavek na fyzikálně založené simulace v oblasti počítačových her a filmového průmyslu. Je možné předpokládat, že tento trend bude v blízké budoucnosti pokračovat a je dobré toho využít pro vývoj aplikací specializovaných na tyto grafické karty.

1.1 Cíl práce

Hlavním cílem mé práce je vytvořit interaktivní simulátor kapalin. K implementaci jsem zvolil metodu Smoothed Particle Hydrodynamics, která je ideální pro využití paralelní architektury grafických karet. Konkrétně se zaměřím na simulaci newtonovských kapalin s nízkou viskozitou (kapaliny podobné vodě). Aplikace by měla být schopná simulovat některé základní scény jako je prolomení přehrady, zaplavení překážek atd. Realistická vizualizace povrchu kapaliny není cílem této práce, kapalina bude zobrazena tak, aby byly odhaleny některé její fyzikální veličiny jako rychlost nebo tlak.

Velkou část mého úsilí věnuji analýze rychlosti aplikace. Výsledky testů budou porovnány s implementacemi ostatních autorů. Dále také demonstruji velký rozdíl ve výkonu mezi implementací pro GPU a CPU a otestuji aplikaci na různých generacích grafických karet.

1.2 Členění práce

V rámci této práce bych chtěl projít cestou, která vede k sestrojení simulátoru kapalin. Práce je členěna takto:

Kapitola 2 – Fyzikální základy simulace obsahuje teoretický úvod do problému simulace kapalin. Jsou zde vysvětleny pojmy jako newtonovská kapaliny a lagrangeovský přistup k simulaci. V kapitole jsou také uvedeny základní formulace Navier-Stokesových rovnic.

Kapitola 3 – Metody simulace tekutin obsahuje shrnutí principů vybraných metod pro simulaci tekutin. V kapitole lze nalézt zástupce eulerovského přístupu simulace, je popsána metoda Lattice Boltzman, sloupcová metoda pro použití v poč. grafice a je nastíněn obecný základ metody Smoothed Particle Hydrodynamics (SPH).

Kapitola 4 - SPH pro newtonské kapaliny pojednává o konkrétní aplikaci metody SPH na lagrangeovskou formulaci dynamiky tekutin. Jsou zde uvedeny konkrétní rovnice a vztahy použité k simulaci newtonských kapalin s nízkou viskozitou.

Kapitola 5– Návrh Řešení a implementace obsahuje návrh řešení a popis zajímavých částí implementace algoritmu SPH.

Kapitola 6 – Výsledky a zhodnocení obsahuje souhrn všech provedených testů aplikace a diskuzi dosažených výsledků.

Kapitola 7 – Závěr obsahuje shrnutí celé práce.

Kapitola 2

Fyzikální základy simulace tekutin

V této kapitole se pokusím vysvětlit pojmy newtonská kapalina a viskozita. Pak se pokusím objasnit rozdíl mezi eulerovským a lagrangeovským přístupem k simulaci kapalin a plynů. Nakonec se budu věnovat vědní disciplíně Výpočetní dynamika tekutin (Computational Fluid Dynamics) a představím rovnice řídící simulace tekutin.

2.1 Newtonské kapaliny a viskozita

Intuitivně si viskozitu lze představit jako jistou formu „hutnosti“ kapaliny. Podle této neformální definice je např. voda méně hutná než med. Voda má tedy nízkou viskozitu a med viskozitu vysokou. Toto intuitivní porozumění je plně dostačující pro běžný život. Odborněji lze říci, že viskozita je měřítkem vnitřního tření v kapalině a představuje odpor kapaliny při aplikaci externí síly.

Většinu kapalin, které jsou běžně za kapaliny považovány (kapaliny podobné vodě), lze velmi dobře aproximovat pomocí modelu navrženého Sirem Isaacem Newtonem:

$$\tau = \mu\left(|\dot{\gamma}|\right)\dot{\gamma}, \tag{2.1}$$

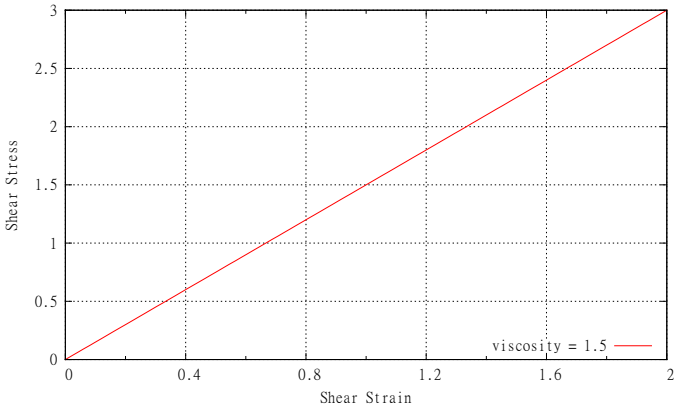
kde τ je míra tečného napětí, $\dot{\gamma}$ je míra deformace ve střihu a μ je dynamická viskozita. Efektivní viskozita μ_{ef} je definována jako poměr tečného napětí a deformace ve střihu:

$$\mu_{ef} = \frac{\tau}{\dot{\gamma}} \tag{2.2}$$

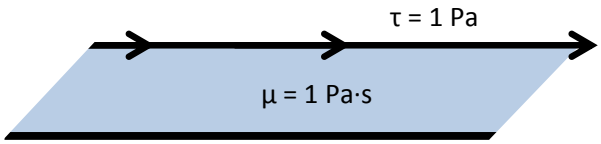
newtonská tekutina je definována jako tekutina jejíž viskozita, tedy poměr mezi příčným napětím a deformací, je konstantní viz obrázek 2.1.

Jednotkou dynamické viskozity $Pa \cdot s$ (vyjádřeno v jednotkách SI: $N \cdot m^{-2} \cdot s$). Newton definoval viskozitu pomocí teoretického experimentu (viz obrázek 2.2), při kterém je kapalina uzavřena mezi dva pláty. Předpokládejme, že tato kapalina má dynamickou viskozitu rovnou jedné paskal-sekundě a že plochy jsem dostatečně velké a mezi pláty a kapalinou je absolutní tření (no-slip condition). Pokud je spodní deska upevněna a na vrchní desku tlačíme s příčným napětím jeden paskal, pak se vrchní plát během jedné sekundy posune o vzdálenost rovnou šířce vrstvy kapaliny.

Je dobré mít na paměti, že dynamická viskozita je ovlivněna teplotou kapaliny. Například voda má v teplotním rozpětí ($0°C$, $100°C$) dynamickou viskozitu rovnou ($1.79 \cdot 10^{-3} Pa \cdot s, 0.28 \cdot 10^{-3} Pa \cdot s$)



Obrázek 2.1: Křivka popisující vztah příčného napětí a rychlosti deformace tekutiny pro kapalinu s dynamickou viskozitou 1.5



Obrázek 2.2: Newtonův teoretický experiment pro znázornění viskozity

2.1.1 Nenewtonské kapaliny

Nenewtonské kapaliny se liší od newtonských kapalin v tom, že mezi příčným napětím a rychlostí deformace neplatí lineární vztah. Viskozita proto nelze použít pro popis těchto látek. Příkladem nenewtonských kapalin jsou např. kečup, zubní pasta, točená zmrzlina atd. Modelováním této skupiny kapalin se nebudu v této práci zabývat.

2.2 Modelování spojitých systému

Podle článku Bridsona [3] existují dva základní způsoby, jak nahlížet na pohybující se spojitou tekutinu. Jedná se o přístup langrangeovský a eulerovský. Tyto rozdílné pohledy na stejný problém jsou pojmenovány po slavných matematicích Lagrangeovi a Eulerovi.

2.2.1 Langrangeovský přístup

Langrangeovský přístup je velmi intuitivní. S modelovaným kontinuem — tedy tekutinou — je zacházeno jako s částicovým systémem. Tekutina je rozdělena na diskrétní částice, které mají přiřazenou příslušnou pozici $\mathbf{x}$ a rychlost $\mathbf{v}$. V krajním případě lze o částicích přemýšlet jako o molekulách tekutiny. Při simulaci tedy sledujeme trajektorii a stavové veličiny všech částic tekutiny.

2.2.2 Eulerovský přístup

Při modelování pomocí eulerovského přístupu, nerozdělujeme kontinuum kapaliny na částice, naproti tomu sledujeme vývoj kapaliny na určitých místech, většinou pevně určených uzly prostorové struktury. U těchto uzlů sledujeme změnu veličin (hustota, rychlost…) tekutiny v čase.

2.3 Výpočetní dynamika tekutin

Konečným cílem oboru výpočetní dynamika tekutin (CFD) je porozumět fyzikálním jevům, které se odehrávají při proudění kapalin. Výpočetní hydrodynamika má široké spektrum aplikací. Nezávisle na konkrétní aplikaci, je v klasické výpočetní hydrodynamice, podle autorů knihy Fundamentals of Computational Fluid Dynamics [17], obecně třeba sledovat následující kroky.

1. Specifikace problému a geometrie
2. Volba řídících rovnic a hraničních podmínek
3. Prostorové rozdělení do mřížky a volba numerické metody
4. Stanovení a interpretace výsledků

Jak je zřejmé z předcházejícího seznamu, autoři Fundamentals of Computational Fluid Dynamics [17] pojednávají pouze o eulerovských metodách využívajících mřížky nebo jiné, složitější, prostorové rozdělení simulované domény. Přesto v současné době nabývají na popularitě numerické metody založené na částicích, a i tato práce se takovou metodou zabývá. Pro metody založené na částicích by bylo možno nahradit bod 3 předcházejícího seznamu takto: Rozdělění objemu kapaliny na částice a volba numerické metody. V následující části se krátce zmíním o rovnicích řídících proudění kapalin.

2.3.1 Navierovy–Stokesovy rovnice

Dynamika tekutin se zabývá pohybem kapalin a plynů, které se při makroskopickém pohledu zdají být spojité. Navierovy-Stokesovy rovnice jsou souborem nelineárních parciálních diferenciálních rovnic, které popisují proudění tekutin. Všechny proměnné jsou považovány za spojité funkce prostorových souřadnic a času. Podle webových stránek Fluid Dynamics and the Navier-Stokes Equations [36] pomocí nich lze modelovat např. počasí, pohyb vzduchu v atmosféře, mořské proudění, proudění v potrubí atd. Zde uvedu eulerovskou i lagrangeovskou formulaci těchto rovnic.

Eulerovská formulace

Navier-Stokesovy rovnice pro stlačitelnou newtonskou kapalinu jsou:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0, \tag{2.3}$$

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{v} = -\frac{1}{\rho}\nabla p + \mathbf{f} + \frac{\mu}{\rho}\nabla^2 \mathbf{v} + \left(\frac{1}{3}\mu + \mu^{\nu}\right) \nabla \left(\nabla \cdot \mathbf{v}\right), \tag{2.4}$$

$$\rho\left(\frac{\partial \epsilon}{\partial t} + \mathbf{v}\cdot\nabla\epsilon\right) - \nabla\cdot(K_H\nabla T) + p\nabla\cdot\mathbf{v} = 0, \tag{2.5}$$

Rovnice 2.3 vyjadřuje zachování hmoty, rovnice 2.4 je rovnicí pohybovou a rovnice 2.5 je vyjádřením zachování energie, kde $\mathbf{v}$ je vektorové pole rychlostí, ρ je hustota tekutiny, μ je viskozita, μ^{ν} je konstanta objemové viskozity $\mathbf{f}$ je vektor externích sil, p je tlak, ϵ je termodynamická vnitřní energie, T je termodynamická teplota a K_H je koeficient vedení tepla. Přičemž ve výpočetní hydrodynamice mají význam zejména rovnice 2.3 a 2.4. Odvození Navier-Stokesových rovnic pro stlačitelnou kapalinu lze nalézt například v článku Desjardinse a Chi-Kuna [9].

Zjednodušení Navier-Stokesových rovnic dosáhneme, pokud budeme uvažovat nestlačitelnou kapalinu (kapaliny, jako je např. voda, jsou v reálu stlačitelné jen málo). Nestlačitelnost kapaliny jednoznačně určuje, že hustota v celém objemu kapaliny je konstantní. Proto se rovnice 2.3 zjednodušší na rovnici

$$\nabla\cdot\mathbf{v} = 0, \tag{2.6}$$

která lze interpretovat, tak, že v nestlačitelné kapalině je divergence vektorového pole rychlosti rovna nule. Protože levá strana rovnice 2.6 se vyskytuje na pravé straně rovnice 2.4, pokud uvažujeme nestlačitelnou kapalinu, celý člen $\left(\frac{1}{3}\mu + \mu^{\nu}\right)\nabla\left(\nabla\cdot\mathbf{v}\right)$ se vynuluje a získáme tvar pohybové rovnice 2.4 pro nestlačitelné kapaliny.

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\cdot\nabla\mathbf{v} = -\frac{1}{\rho}\nabla p + \mathbf{f} + \frac{\mu}{\rho}\nabla^2\mathbf{v}. \tag{2.7}$$

Na levé straně rovnice je zrychlení. To se skládá ze dvou částí, ze zrychlení nestálého $\frac{\partial \mathbf{v}}{\partial t}$, které závisí na čase t a ze zrychlení konvektivního závisejího na pozici $\mathbf{v}\cdot\nabla\mathbf{v}$. Pravá strana představuje příspěvky jednotlivých sil ke zrychlení kapaliny v určitém čase a místě v prostoru. Výraz $-\nabla p$ vyjadřuje zrychlení způsobené gradientem tlakového pole, záporné znaménko vyjadřuje skutečnost, že síla působí z míst s větším tlakem do míst s tlakem menším. $\mathbf{f}$ je zrychlení způsobené externími silami, jako je gravitace. Výraz $\frac{\mu}{\rho}\nabla^2\mathbf{v}$ představuje vliv viskozity na kapalinu, tedy, jak uvádí Harris [13], její „schopnost odolávat toku“ .

Lagrangeovská formulace

Rovnice zachování hmoty v lagrangeovské formulaci vypadá takto:

$$\frac{d\rho}{dt} = -\rho\nabla\cdot\mathbf{v}, \tag{2.8}$$

Zachování hybnosti pro tok nestlačitelné newtonské kapaliny je formulováno následovně:

$$\frac{d\mathbf{v}}{dt} = -\frac{1}{\rho}\nabla p - \frac{\mu}{\rho}\nabla^2\mathbf{v} + \mathbf{f} \tag{2.9}$$

2.3.2 Eulerovy rovnice

Leonhardt Euler v roce 1755 ve svém díle o aplikované matematice formuloval obecné principy zachování hybnosti a hmoty pro tekutiny [26]. Tyto rovnice jsou dnes známy jako Eulerovy rovnice pro tekutiny:

$$\frac{\partial \rho}{\partial t} + \nabla\cdot\rho\mathbf{v} = 0, \tag{2.10}$$

$$\rho \left(\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla) \mathbf{v} \right) = -\nabla p + \rho \mathbf{F}. \tag{2.11}$$

Jak je zřejmé z předchozí části 2.3.1 popisující Navier-Stokesovy rovnice , Eulerova rovnice 2.11 je speciálním případem rovnice 2.7, ze které je odebrán vliv viskozity. Z toho vyplývá, že Eulerovy rovnice je možné použít pro simulaci kapalin s nízkou viskozitou. Peralta-Fabi [26] popisuje problémy, které tyto rovnice při simulaci způsobují.

Kapitola 3

Metody simulace tekutin

V této kapitole udělám stručný přehled vybraných metod používaných pro simulaci tekutin a to jak v inženýrských aplikacích, tak v počítačové grafice. U každé metody se pokusím vysvětlit základní principy. V závěru kapitoly se budu věnovat obecným charakteristikám metody Smoothed Particle Hydrodynamics, jejíž aplikaci na simulaci newtonských kapalin popíšu v následující kapitole.

3.1 Eulerovské metody založené na mřížce

Chceme-li simulovat chování tekutin, musíme znát fyzikální reprezentaci stavu kapaliny v daném okamžiku. Nejdůležitější kvantita vyžadující reprezentaci je rychlost tekutiny. Rychlost určuje, jak tekutina pohybuje sama sebou a s věcmi, které jsou v ní ponořeny. Protože se rychlost kapaliny mění v čase i v prostoru, je v eulerovských metodách modelována jako vektorové pole.

Toto pole je zpravidla vázáno na nějakou prostorovou strukturu. Jednoduchým případem je trojrozměrná karteziánská mřížka, jakou například používá Stam [28], její dvojrozměrná obdoba v podání Harrise [13] je na obrázku 3.1. Komplikovanější modely mohou využívat různé adaptivní mřížky umožňující zjemnění přesnosti simulace v oblastech zájmu. Losasso, Gibou a Fedkiw [18] např. použili k simulaci vody a kouře adaptivní oktantový strom. Využití složitějších prostorových struktur ovšem výrazně zpomaluje výpočet a ztěžuje implementaci. V následujících odstavcích popíši řešení Navier-Stokesových rovnic v dvojrozměrném prostoru použité Harrisem [13].

V Navier-Stokosově rovnici 2.7 je Laplaceův operátor aplikován na vektorové pole $\nabla^2\mathbf{v}$, jedná se o zjednodušení, operátor je aplikován na jednotlivé skalární složky vektorového pole. V dvojrozměrném případě tedy z rovnice 2.7 získáváme dvě rovnice 3.1 a 3.2

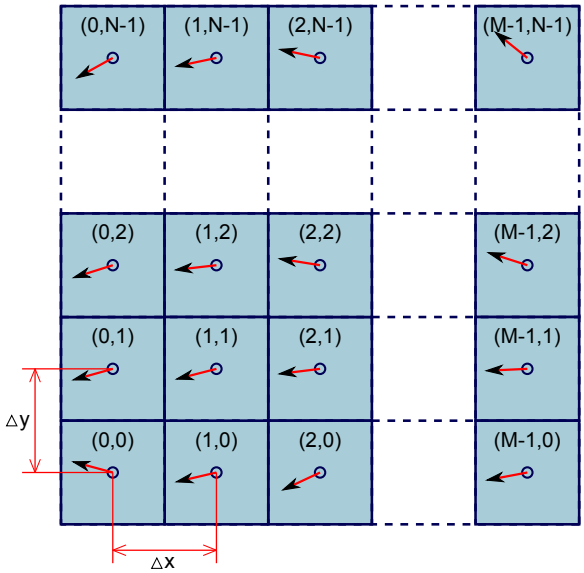
$$\frac{\partial u}{\partial t} + (\mathbf{v} \cdot \nabla)\, u = -\frac{1}{\rho}\nabla p + \mathbf{F} + \frac{\mu}{\rho}\nabla^2 u \tag{3.1}$$

$$\frac{\partial v}{\partial t} + (\mathbf{v} \cdot \nabla)\, v = -\frac{1}{\rho}\nabla p + \mathbf{F} + \frac{\mu}{\rho}\nabla^2 v, \tag{3.2}$$

kde u a v jsou skalární složky vektorového pole rychlosti $\mathbf{v}$. Spolu s rovnicí (2.6) máme tedy tři rovnice a tři neznámé (u, v a p).

Algoritmus pro řešení Navier-Stokecosých rovnic použitý Harrisem [13] je založen na následující rovnici

$$S(\mathbf{v}) = P \circ F \circ D \circ A(\mathbf{v}), \tag{3.3}$$



Obrázek 3.1: Vektorové pole rychlosti kapaliny vázané na dvojrozměrnou karteziánskou mřížku

kde S je operátor reprezentující řešení — tedy aplikujeme-li S na pole rychlostí $\mathbf{v}(t)$, dostaneme pole v dalším časovém kroku $\mathbf{v}(t + \delta t)$. Operátor A představuje vliv advekce na rychlost, D vliv viskozity (difúze rychlosti), F je vliv externích sil a P je zvláštní operátor navržený Stamem [28]. P je založen na faktu, že uvažujeme neztlačitelnou kapalinu, kapalinu mající nulovou divergenci pole rychlosti. Po aplikování operátorů A, D a F na vektorové pole $\mathbf{v}$ ovšem dostaneme pole $\mathbf{w}$, které má nenulovou divergenci. Podle Helmholtz-Hodge dekompozičního teorému

$$\mathbf{w} = \mathbf{v} + \nabla p \tag{3.4}$$

lze vektorové pole $\mathbf{w}$, definované na prostoru $\mathbf{D}$, který má diferenciovatelné hranice s normálou $\mathbf{n}$, rozložit na vektorové pole $\mathbf{v}$ s nulovou divergencí a gradient skalárního pole — v našem případě tlaku. Odvození a důkaz lze nalézt v knize Chorina a Marsdena [5]. Aplikací operátoru divergence na rovnici 3.4 získáme rovnici

$$\nabla \cdot \mathbf{w} = \nabla \cdot \mathbf{v} + \nabla^2 p \tag{3.5}$$

a když zvážíme, že $\mathbf{v}$ má nulovou divergenci, dostaneme Poissonovu rovnici

$$\nabla^2 p = \nabla \cdot \mathbf{w} \tag{3.6}$$

Pokud tedy známe $\mathbf{w}$ můžeme vypočítat p a pak podle rovnice 3.4 i pole $\mathbf{v}$. Výše zmíněný operátor P z rovnice 3.3 je pak definován tak, že promítne $\mathbf{w}$ na svůj komponent s nulovou divergencí:

$$P(\mathbf{w}) = P(\mathbf{v}) = \mathbf{v}. \tag{3.7}$$

Po aplikaci operátoru P na rovnici (3.4) pak tedy dostaneme, že $P(\nabla p) = 0$. Po aplikaci operátoru P na Navier-Stokesovu rovnici (2.7) a po zvážení, že derivát $\frac{\partial \mathbf{v}}{\partial t}$ má nulovou divergenci, získáme rovnici

$$\frac{\partial \mathbf{v}}{\partial t} = P\left(-(\mathbf{v} \cdot \nabla)\mathbf{v} + \mathbf{F} + \frac{\mu}{\rho}\nabla^2 \mathbf{v}\right). \tag{3.8}$$

Tato rovnice odpovídá rovnici (3.3), která byla vyjádřena za pomocí operátorů vyjadřujících jednotlivé komponenty rovnice (3.8). To odpovídá následujícímu algoritmu zapsanému v notaci jazyka C (převzato z [13]).

Aplikace operátorů pro řešení N-S rovnice

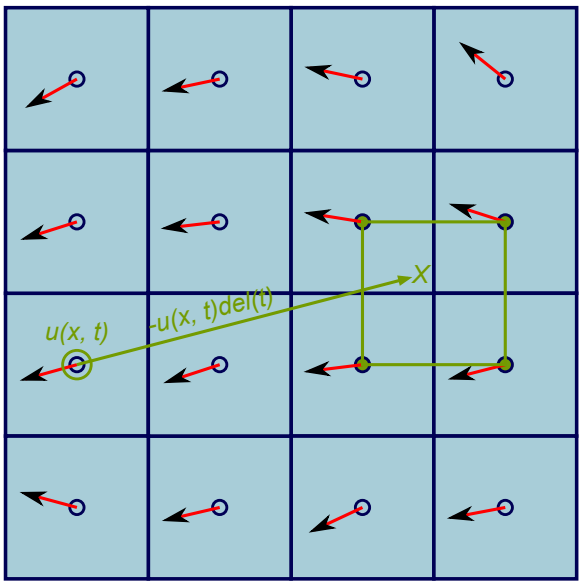
```
// Apply the first 3 operators A, D, F
v = advect(v);
v = diffuse(v);
v = addForces(v);
// Now apply the projection operator P to the result.
p = computePressure(v);
v = subtractPressureGradient(v, p);
```

Co se týče výpočtu advekce a difúze, Stam [28] ukazuje, že použití dopředných integračních metod (Eulerova, Runge-Kutta…) není vhodné, protože mohou být pro velké časové kroky ($\mathbf{v}\delta t > gridSize$) nestabilní. Eulerova metoda pracuje takto:

$$q(t + \delta t) = q(t) + \mathbf{v}(t)\delta t. \tag{3.9}$$

Místo toho využívá přístup obrácený. Pro určité místo v prostoru x a čas $t+\delta t$ se „podívá“ve směru opačné rychlosti a zkopíruje kvantitu q z místa $x-\mathbf{v}(x,t)\delta t$ do x. Dohromady metoda pracuje podle následujícího schéma a je znázorněna na obrázku 3.2.

$$q(x, t + \delta t) = q(x - \mathbf{v}(x, t)\delta t, t). \tag{3.10}$$



Obrázek 3.2: Znázornění zpětné metody výpočtu veličin u mřížkové metody.

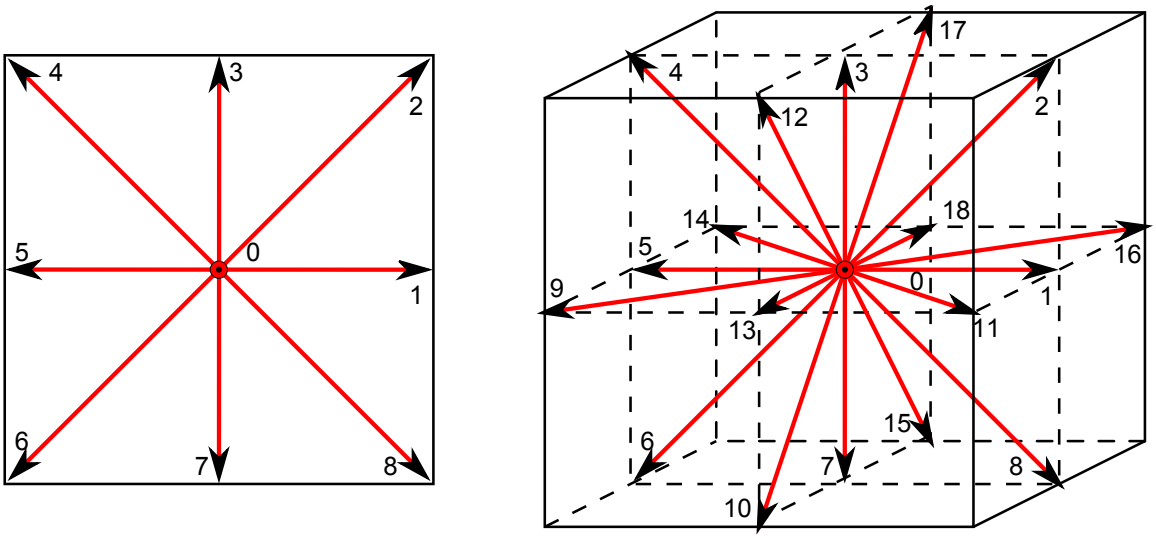
3.2 Metoda Lattice Boltzman

Podle článku pánů Succiho, Sbragaglia a Ubertiniho [30] existují dva hlavní druhy popisu tekutin. Jedná se o popis makroskopický a mikroskopický (spojitý a atomický). Při popisu

spojitém je tekutina popsána z hlediská prostor vyplňujících polí jako je hustota, rychlost, tlak, které se v prostoru spojitě liší. Takový popis je pro většinu aplikací zcela dostačující zejména proto, že kapalina často bývá právě těmito veličinami definována. Nicméně je známo, že tekutiny jsou složené z kolekce jednotlivých atomů a molekul, jejichž diskrétní povaha se projeví při měřítcích menších než řádově nanometry.

Existuje třetí, méně známá, úroveň popisu, ve které jsou tekutiny popsány z hlediska funkce hustoty pravděpodobnosti $f(\mathbf{r}, \mathbf{v}, t)$, která určuje pravděpodobnost, že se daná částice nachází na daném místě v prostoru $\mathbf{r}$ a v čase t s danou rychlostí $\mathbf{v}$. jak uvádí Wagner [34], metoda původně využívala diskrétní částice, které se pohybovaly po mřížce, teprve později byla rozšířena k použití funkce hustoty pravděpodobnosti.

V metodě Lattice-Boltzmann je prostor rozdělen do pravidelné mřížky, např. krychlové ve 3D, čtvercové nebo šestiúhelníkové ve 2D. V každém bodě mřížky je hodnota funkce pravděpodobnosti pro každou částici reprezentována reálným číslem, které je rovné očekávanému počtu identických částic v každém z dostupných stavů částice. V nejjednodušším případě je každý stav reprezentován rychlostí částice limitované na diskrétní množství (obrázek 3.3). Během každého simulačního kroku částice přeskočí do nejbližšího bodu mřížky po směru svých rychlosti, kde se střetnou s ostatními částicemi ve stejné pozici. Výsledek těchto kolizí se určí pomocí Boltzmannovy rovnice pro nové rozložení částic a upraví se distribuční funkce částic. V kubické mřížce je distribuční funkce pro jednu částici v jednom místě prostoru určena jako $f_i(\mathbf{x}t)$, očekávaný počet částic v místě $\mathbf{x}$ v čase t, které se pohybují ve směru vektoru rychlosti e_i, kde každá z hodnot indexu i reprezentuje povolený směr rychlosti, jak je znázorněno na obrázku 3.3. Obrázek ukazuje 8 směrů rychlosti pro dvojrozměrnou variantu a 19 směrů rychlosti pro trojzrozměrnou variantu. Existuje ovšem více možností, např 15 směrů rychlostí pro 3D jak použili např. Chen, Doolen a Egert [4]. Pro trojrozměrný případ s 18 směry rychlostí a nulovou rychlostí tedy popisuje distribuční funkci částice 19 čísel.



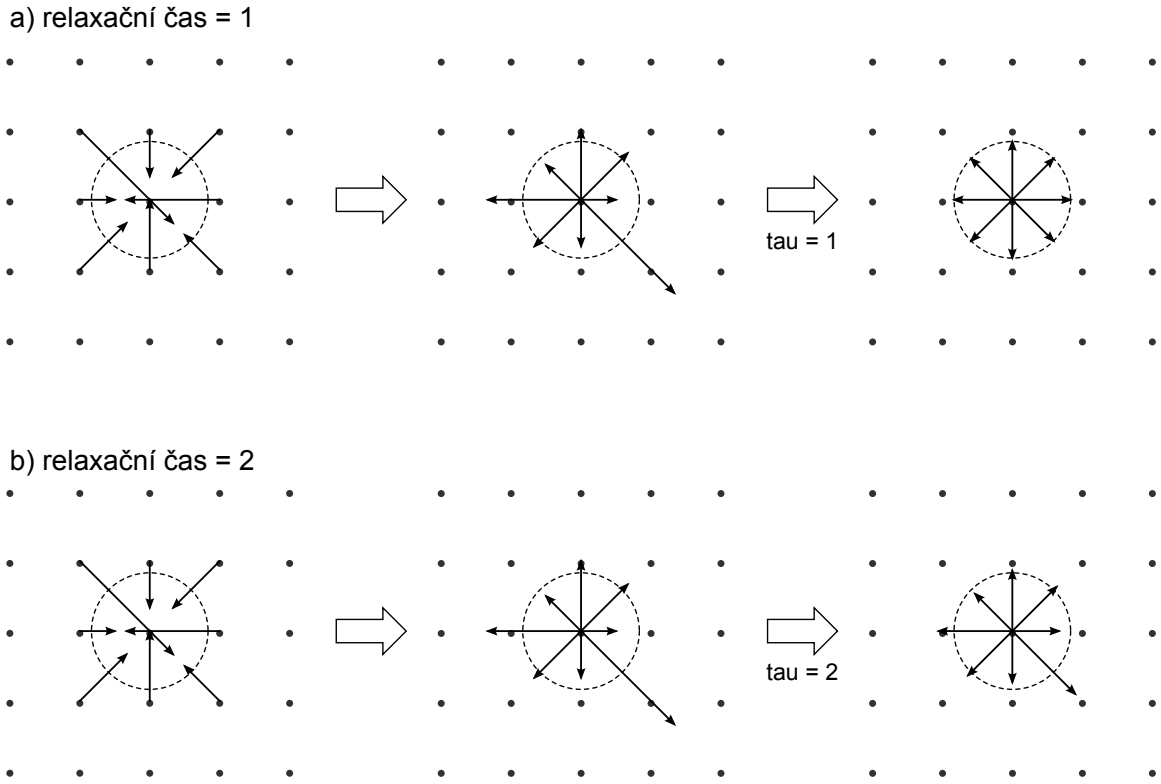
Obrázek 3.3: Vektory rychlosti pro typickou diskretizaci metodou Lattice-Boltzman ve 2D a ve 3D

První operace v každém časovém kroku δt je posunutí částice do jiného bodu mřížky ve směru rychlosti. Model popisovaný Chenem, Doolenem a Egertem [4] (15 vektorů rychlosti) má tedy pouze tři diskrétní velikosti rychlosti — nulová rychlost, c a $\sqrt{3}c$. Dalším krokem

je simulace kolizí. Pravidla kolizí jsou nastavena tak, aby kolize ponechávaly sumu f_i nepozměněnou. Také jsou volena, aby zachovávala celkovou energii a hybnost v jednotlivých bodech mřížky. Výsledek kolize je jednoduše aproximován za předpokladu, že hybnost interagujících částic bude konstantní rychlostí redistribuována směrem k rovnovážnému stavu rozložení hustoty pravděpodobnosti f_i^{eq}. Vývoj distribuční funkce pro jednu částici je dán vztahem

$$f_i\left(\mathbf{x}+\mathbf{e}_i, t+1\right) = f_i\left(\mathbf{x}, t\right) - \frac{f_i\left(\mathbf{x}, t\right) - f_i^{eq}\left(\mathbf{x}, t\right)}{\tau}, \tag{3.11}$$

kde $i = 0, 1, ..., 14$. $\frac{1}{\tau}$ je rychlost přibližování k rovnovážnému stavu — parametr τ ovlivňuje viskozitu modelu tekutiny, viz obrázek 3.4.



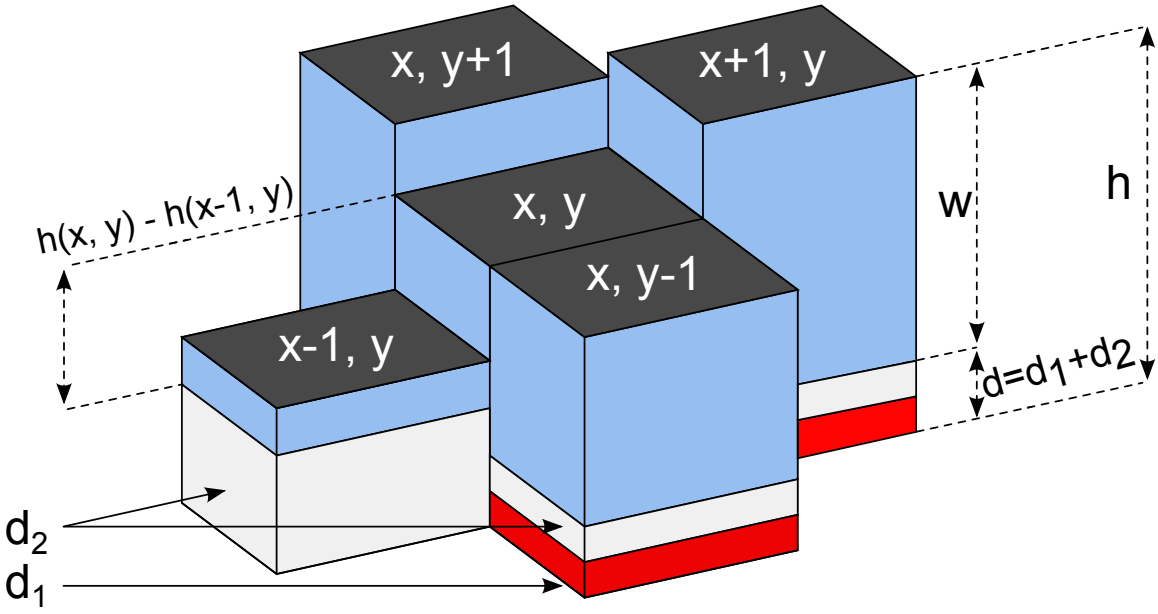
Obrázek 3.4: Relaxační proces po kolizi po uplynutí jednoho časového kroku. Při absenci externích sil je rovnovážný stav rozložení rychlosti takový, že rychlosti ve všech směrech jsou stejné. Obrázek zachycuje dva výpočetní kroky během jednoho kroku simulace. Nejprve se přicházející rychlosti shromáždí v bodě mřížky — sousedící částice přeskočí po směru své rychlosti. V druhém kroku se změní funkce distribuce, podle pravidla relaxace v závislosti na parametru τ, na výstupní distribuci, která je blíže rovnovážnému stavu. Při hodnotě parametru $\tau = 1$ se přicházející rozložení pravděpodobosti změní během jednoho kroku na rovnovážný stav.

Succi [29] uvádí, že metody typu Lattice-Boltzmann jsou obvzláště vhodné pro simulování tekutin na velmi členitém povrchu a pro simulaci vícefázových toků.

3.3 Sloupcová metoda

Sloupcová metoda je velmi jednoduchá metoda simulace kapalin pro účely počítačové grafiky. Je využívána pro rychlý (realtime) výpočet v rozsáhlých scénách a pro simulaci eroze materiálu jako např v článku Interactive terrain modeling using hydraulic erosion [33].

Při simulaci je vytvořena dvojrozměrná síť, která pokrývá „půdorys“ simulované kapaliny. Kapalina je reprezentována výškou sloupce v jednotlivých čtvercích sítě, viz obrázek 3.5.



Obrázek 3.5: Znázornění reprezentace kapaliny sloupcovou metodu, d_1 a d_2 jsou různé podkladové materiály

Metoda pracuje na principu vyrovnávání hydrostatického tlaku. Rozdíl hydrostatického tlaku $\delta P_{i,j}$ mezi sloupcem (x, y) a jeho sousedním sloupcem $(x \pm 1, y \pm 1)$ je

$$\delta P_{i,j}(x, y) = \rho g \delta h_{i,j}(x, y), \tag{3.12}$$

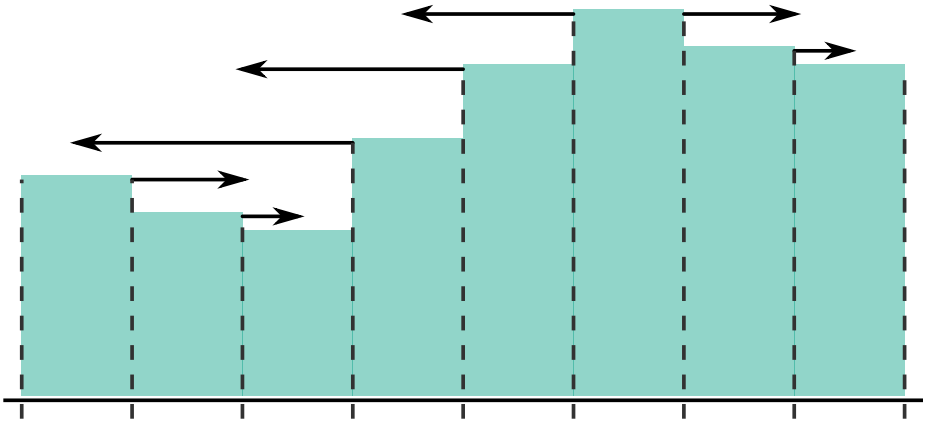
kde ρ je hostota kapaliny, g je gravitační zrychlení a $\delta h_{i,j}(x, y)$ je rozdíl výšky příslušných vodních sloupců. Zrychlení kapaliny tekoucí přes virtuální rouru v důsledku „snahy“ kapaliny o vyrovnání tlaku na obou koncích je

$$a_{i,j}(x, y) = \frac{\delta P_{i,j}(x, y)}{\rho l}, \tag{3.13}$$

kde l je délka roury (vzdálenost dvou sloupců). Nová výška vodního sloupce je pak rovna:

$$w^{t+\delta t}(x, y) = w^t(x, y) + \frac{\delta t}{l^2} \sum_{i,j} \left[f^t_{i,j}(x - i, y - j) - f^t_{i,j}(x, y) \right] \tag{3.14}$$

Vlček ve své diplomové práci [32] používá metodu poněkud jednodušší. K vyrovnávání výšek vodních sloupců používá pouze rozdíl hladin. Pro dvojrozměrný případ je to znázorněno na obrázku 3.6.



Obrázek 3.6: Znázornění velikosti rychlosti přelévání vody v buňkách v závisloti na výšce okolních vodních sloupců

3.4 Smoothed Particle Hydrodynamics

Metodu Smoothed Particle Hydrodynamics[1] vyvinuli téměř současně v roce 1977 Lucy [19] a Gingold s Monaghanem [10] pro využití v astrofyzice. Od té doby, jak píše Dalrymple [8], je SPH jedním z hlavních nástrojů pro simulaci vývoje galaxií, kolizí galaxií a srážek bolidů. V roce 1994 Monaghan ukázal, že SPH lze využít i v hydrodynamice a to s výhodou pro modelování tekutin s volným povrchem, u kterých není na povrchu potřeba žádné speciální zacházení.

Metoda SPH je metoda interpolační a je založena na následujícím integrálu

$$u(\mathbf{s},t) = \int_v W(\mathbf{s}-\mathbf{x},h)u(\mathbf{x},t)dv, \tag{3.15}$$

kde integrál je přes celou doménu v a dv je ve třech rozměrech $dxdydz$, $dxdy$ ve dvou a dx v jednom rozměru. Definiční obor váhové funkce $W(\mathbf{s}-\mathbf{x},h)$ je určen velikostí parametru h (vyhlazovací délka) a je, jak uvádí Dalrymple [8] roven kouli (kruh ve 2D) o poloměru $2h$. V jednodušších aplikacích je h konstantní, ve složitějších aplikacích může být h funkcí pozice $\mathbf{x}$ a času t. Využití proměnných vyhlazovacích délek např. diskutují ve svém článku Nelson a Papaloizou [24].

Váhová funkce $W(\mathbf{s}-\mathbf{x},h)$ bývá často nazývána jádro. V metodě SPH jsou na jádro kladeny speciální požadavky. Jedním z nich je, že jádro se musí chovat jako funkce delta $\delta(\mathbf{x})$, která je definována takto:

$$\int_v f(x)\delta(\mathbf{x}-c)dx = f(c) \tag{3.16}$$

$$\int_v \delta(\mathbf{x}-c)dx = 1; c \in \Re \tag{3.17}$$

Příklad Diracovy funkce v jedné dimenzi uvádí např. Liu a Li [15]

$$\delta(x) = \lim_{\epsilon\to 0}\begin{cases} 0, & x < -\epsilon/2 \\ 1/\epsilon, & -\epsilon/2 < x < \epsilon/2 \\ 0, & x > \epsilon/2 \end{cases} \tag{3.18}$$

[1] dále jen SPH

Pokud jádro vykazuje chování, jako delta funkce, nazývá se reprodukující jádro. Další vlastností jádra je kompletnost (konzistence), dosadíme-li do rovnice (3.15) $u(\mathbf{x}, t) = 1$, dostaneme:

$$\int_v W(\mathbf{s} - \mathbf{x}) dv = 1, \tag{3.19}$$

což znamená, že rovnice správně reprodukuje konstanty. Monaghan v roce 1992 [20] zavedl další podmínku pro jádro:

$$\lim_{h \to 0} W(\mathbf{x}, h) = \delta(\mathbf{x}) \tag{3.20}$$

Abychom mohli tyto rovnice použít k numerickým výpočtům je třeba integrály aproximovat pomocí sumy příspěvků jednotlivých částic v simulované doméně:

$$u(\mathbf{s}, t) \approx \sum_j W(\mathbf{s} - \mathbf{x}_j, h) u_j \Delta v_j \tag{3.21}$$

$$\sum_j W(\mathbf{s} - \mathbf{x}_j, h) \Delta v_j = 1, \tag{3.22}$$

kde Δv_j je inkrementální objem(obsah ve 2D) spojený s jednou částicí. V metodě SPH je hmotnost každé $j$té částice m_j neměnná, zatímco hustota v okolí částice ρ_j se může měnit. u_j, m_j a ρ_j nejou funkcí času a m_j není funkcí místa v prostoru. Tím pádem můžeme nahradit $\Delta v_j = m_j / \rho_j$ a přepsat rovnici(3.21) na:

$$u(\mathbf{s}, t) \approx \sum_j \frac{m_j}{\rho_j} W(\mathbf{s} - \mathbf{x}_j, h) u_j(t). \tag{3.23}$$

3.4.1 Vyhlazovací funkce

Jedním z hlavních problémů SPH je volba funkce jádra. Jádro musí splňovat různé podmínky v závislosti na konkrétní aplikaci metody. Na obrázku 3.7 je znázorněno gaussovské jádro, které určuje příspěvky jednotlivých částic v dvojrozměrném prostoru.

V aplikacích simulujících tekutiny pomocí metody SHP je potřeba počítat gradient nebo laplacián vektorového resp. skalárního pole. Velkou výhodou SPH je, že tyto derivace se ve výsledku týkají pouze jádra. Zderivujeme-li obě strany rovnice (3.23), dostaneme:

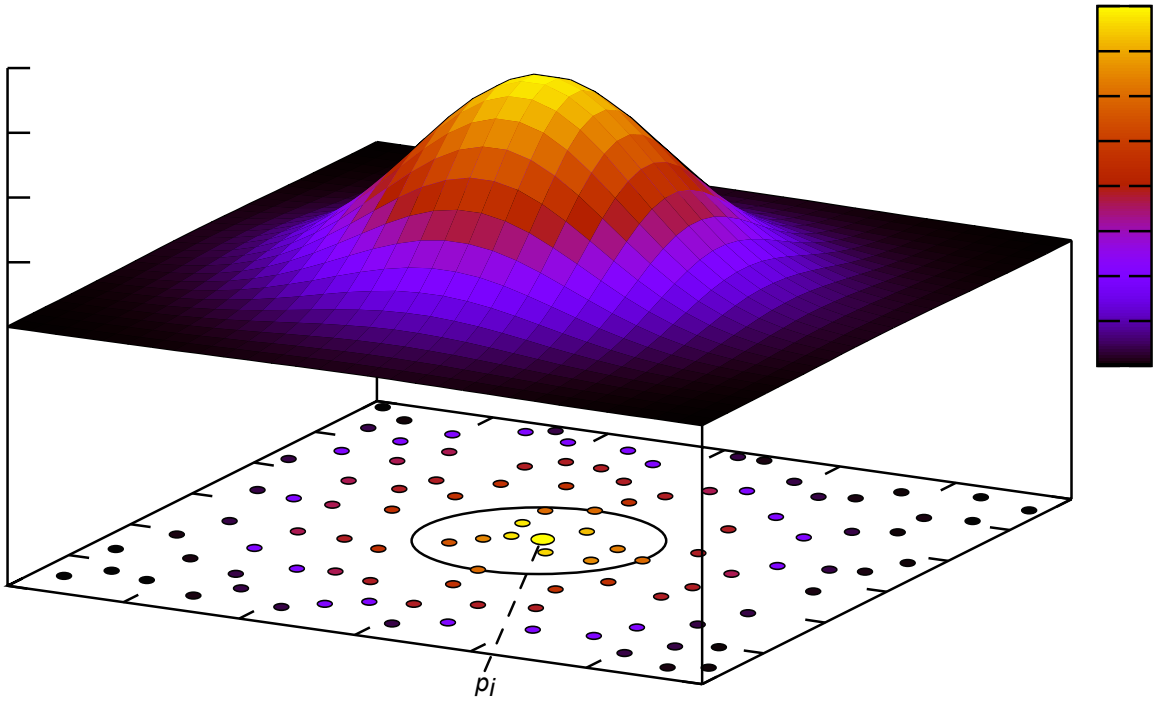
$$\nabla u(\mathbf{s}, t) \approx \nabla \sum_j \frac{m_j}{\rho_j} W(\mathbf{s} - \mathbf{x}_j, h) u_j(t). \tag{3.24}$$

Derivace se týká pouze jádra, nic jiného není na pravé straně závislé na x_j. Proto získáme

$$\nabla u(\mathbf{s}, t) \approx \sum_j \frac{m_j}{\rho_j} u_j(t) \nabla W(\mathbf{s} - \mathbf{x}_j, h) \tag{3.25}$$

a analogicky pro laplacián

$$\nabla^2 u(\mathbf{s}, t) \approx \sum_j \frac{m_j}{\rho_j} u_j(t) \nabla^2 W(\mathbf{s} - \mathbf{x}_j, h). \tag{3.26}$$



Obrázek 3.7: Znázornění gausssovského jádra ve dvojrozměrném prostoru. Na obrázku je znázorněn příspěvek jednotlivých částic k působení na částici p_i. Velikost příspěvku je znázorněna na barevné stupnici.

Korekce laplaciánu

Základní formulace výpočtu laplaciánu metodou SPH se při určitých okolnostech ukázala jako nestabilní, proto pánové Shalo a Lo [27] navrhli opravu nestabilit pomocí alternativní formulace laplaciánu (využito při výpočtu viskózních sil):

$$\nabla \cdot \left(\frac{1}{\rho} \nabla A \right) = \sum m_j \frac{8}{(\rho_i + \rho_j)^2} \frac{(A_i - A_j) \cdot \nabla W(\mathbf{r}_{ij}, h)}{|\mathbf{r}_{ij}|^2 + \eta^2}, \tag{3.27}$$

kde η je malá konstanta, která udržuje jmenovatel různý od nuly, většinou má hodnotu $0.1h$.

3.4.2 Aspekty SPH při simulaci kapalin

Použití metody SPH při simulaci kapalin je velmi atraktivní z několika důvodů. V kontextu real-time simulací se jedná zejména o možnost poměrně snadné paralelizace algoritmu. Mezi další výhody patří například metodou dané zachování hmotnosti. Oproti eulerovským metodám jsou snadné výpočty kolizí s různorodými překážkami, protože není nutné používat mřížky s vysokým rozlišením.

Pravděpodobně největší nevýhodou metody je nesnadná a časově náročná extrakce hladkého povrchu. Pro získání vizuálně realistických výsledků je také třeba poměrně hodně částic. Jak efektivně vizualizovat kapalinou simulovanou metodou SPH popisuje Harada et. al [12].

Dobré shrnutí použití metody SPH lze nalézt například v článku pánů Clearyho a Prakashe [6].

Kapitola 4

SPH pro newtonské kapaliny s nízkou viskozitou

V eulerovské formulaci jsou kapaliny popsány polem rychlostí $\mathbf{v}$, polem hustot ρ a tlakovým polem p. Vývoj těchto veličin v čase je dán pomocí rovnic(2.3) a (2.7).

Jak uvádí Müller [23], použití částic namísto stacionární mřížky výrazně zjednodušší tyto rovnice. Protože počet částic je konstantní a každá částice si zachovává svou hmotnost m, zachování hmotnosti je zaručeno a rovnici (2.3) lze úplně vynechat. Další zjednodušení také vyplývá z částicové povahy simulace. Výraz $\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v}$ na levé straně rovnice (2.7) lze nahradit substanciálním derivátem $\frac{D\mathbf{v}}{Dt}$. Protože se částice pohybují zároveň s tekutinou, substanciální derivát pole rychlosti přechází v derivaci rychlosti v čase, to znamená, že konvektivní výraz není pro simulaci částicovým systémem potřeba a můžeme jej vypustit. Po zmíněných úpravách a vynásobení rovnice (2.7) hustotou ρ dostaneme

$$\frac{d\mathbf{v}}{dt} = \frac{1}{\rho}\left(-\nabla p + \mu \nabla^2 \mathbf{v} + \mathbf{F}\right). \tag{4.1}$$

Na pravé straně rovnice jsou tři výrazy a každý odpovídá silovému poli. $-\nabla p$ modeluje tlakové pole, $\mu \nabla^2 \mathbf{v}$ modeluje viskozitu a $\mathbf{F}$ externí síly. Součet těchto tří silových polí určuje celkové zrychlení částice, které lze pro částici i vyjádřit za pomoci celkové síly takto:

$$\mathbf{a_i} = \frac{d\mathbf{v}}{dt} = \frac{\mathbf{F}_i}{\rho_i} \tag{4.2}$$

4.1 Hustota

Z definice metody SPH vyplývá, že pro její aplikaci na jednotlivé výrazy pravé strany rovnice 4.1, musíme znát hmotnost částice m_i, ta je pro částici konstantní a ve většině aplikací stejná pro všechny částice. Dále musíme znát hustotu ρ_i asociovanou s částicí i. Hustota je spojitou veličinou pro kapalinu a lze ji vypočítat z rovnice(3.23)

$$\begin{aligned} \rho_i &= \sum_{j=1}^{N} \rho_j \frac{m_j}{\rho_j} W(\mathbf{s} - \mathbf{x}_j, h) \\ &= \sum_{j=1}^{N} m_j W(\mathbf{s} - \mathbf{x}_j, h). \end{aligned} \tag{4.3}$$

Jádro použité v této rovnici je popsáno v části 4.4.2.

4.2 Vnitřní síly

Jedná se o síly vznikající v samotné kapalině. Tedy o sílu tlakového pole a síly viskózní. Vliv těchto sil je patrný z rovnice (4.1).

4.2.1 Nestlačitelnost a tlak

K výpočtu tlakového působení na částici je třeba znát tlak v místě všech částic. Ten lze vyjádřit více způsoby. Nejdříve je ovšem třeba se postarat o to, aby kapalina svým chováním alespoň přibližně splňovala kritérium nestlačitelnosti.

Nestlačitelnost

Jako jedno z prvních řešení se nabízí vyřešit přímo Poissonovu rovnici

$$\nabla^2 P = \rho \frac{\nabla \mathbf{v}}{\nabla t}, \tag{4.4}$$

čímž lze téměř dosáhnout nestlačitelnosti. Výpočet je ovšem velmi náročný a přestože použití Poissonovy rovnice umožňuje použít delší časový krok, jsou celkové výpočetní nároky větší než u použití některé z variant stavové rovnice viz níže.

Müller [23] navrhuje použití upravené stavové rovnice pro ideální plyny

$$pV = nRT, \tag{4.5}$$

kde p je tlak, $V = \frac{1}{\rho}$ je objem na jednotku hmotnosti, R je plynová konstanta a T je termodynamická konstanta. Pro neměnnou teplotu je pravá strana rovnice konstantní a můžeme ji nahradit konstantou k. Můžeme tedy dosáhnout tvaru rovnice

$$\begin{aligned} pV &= k \\ p\frac{1}{\rho} &= k \\ p &= k\rho. \end{aligned} \tag{4.6}$$

Protože se ale jedná o rovnici pro ideální plyn, výsledná síla bude vždy odpudivá, proto Müller [23] používá úpravu rovnice

$$p = k(\rho - \rho_0), \tag{4.7}$$

kde ρ_0 je klidová hustota. Harada et al. [12] používá verzi rozšířenou o klidový tlak:

$$p = p_0 + k(\rho - \rho_0). \tag{4.8}$$

Poněkud složitější přístup používá Dalrymple [8]. Stavová rovnice v jeho podání vypadá takto

$$p = B\rho_s \left[\left(\frac{\rho}{\rho_s} \right)^{\gamma} - 1 \right], \tag{4.9}$$

kde B je konstanta, ρ_s je referenční hustota, většinou měřená na volném povrchu, kde je nulový tlak. γ je polytropická konstanta mezi 1 a 7. Podle Morrise, Foxe a Zhua [22] je $\gamma = 7$ vhodné pro simulace oceánů a $\gamma = 1$ pro toky s nízkým Reynoldsovým číslem. Výhodou řešení stavových rovnic je, že se vyhneme řešení Poissonovy rovnice pro tlak. K výpočtu

konstanty B je potřeba znát rychlost zvuku v kapalině. Druhá mocnina rychlosti zvuku je dánna derivátem tlaku podle hustoty:

$$C^2(\rho) = \frac{\partial p}{\partial \rho} = B\gamma \left(\frac{\rho}{\rho_s}\right)^{\gamma-1}, \tag{4.10}$$

z čehož vyplývá, že konstanta B odpovídá druhé mocnině rychlosti zvuku na povrchu dělené polytropickou konstantou C^2/γ. Dalrymple uvádí, že volba reálné rychlosti zvuku pro kapaliny vede k příliš krátkým časovým krokům. Monaghan [10] empiricky ukázal, že tato rychlost může být uměle zmenšena, aniž by bylo chování kapaliny změněno, pokud bude rychlost zvuku alespoň 10krát větší než předpokládaná maximální rychost toku kapaliny.

Zajištění nestlačitelnosti kapaliny pomocí stavové rovnice není ideální, protože stavová rovnice je v zásadě tvrdá rovnice, která může do systému zavést nestability. Při používání stavových rovnic (v této práci jsem použil rovnici(4.8)) je třeba mít tyto věci na paměti a pečlivě vyvážit velikost simulačního kroku v závislosti na konstantě tuhosti. Při interaktivní simulaci, kde hlavním cílem je rychlost — a tedy velikost simulačního kroku — je možné přistoupit na větší stlačitelnost kapaliny, která umožní delší časový krok.

Tlak

Podle rovnice (4.1) má tlak v kapalině za následek pohyb částic ve směru opačném k gradientu tlakového pole, jako je znázorněno na obrázku 4.1. Sílu působící na částice v důsledku tlakového pole f_i^{press} lze tedy vyjádřit vztahem

$$f_i^{press} = -\frac{1}{\rho_i}\nabla p(\mathbf{x}_i). \tag{4.11}$$

Aplikací rovnice (3.23) získáme vztah pro výpočet tlakové síly pomocí metody SPH

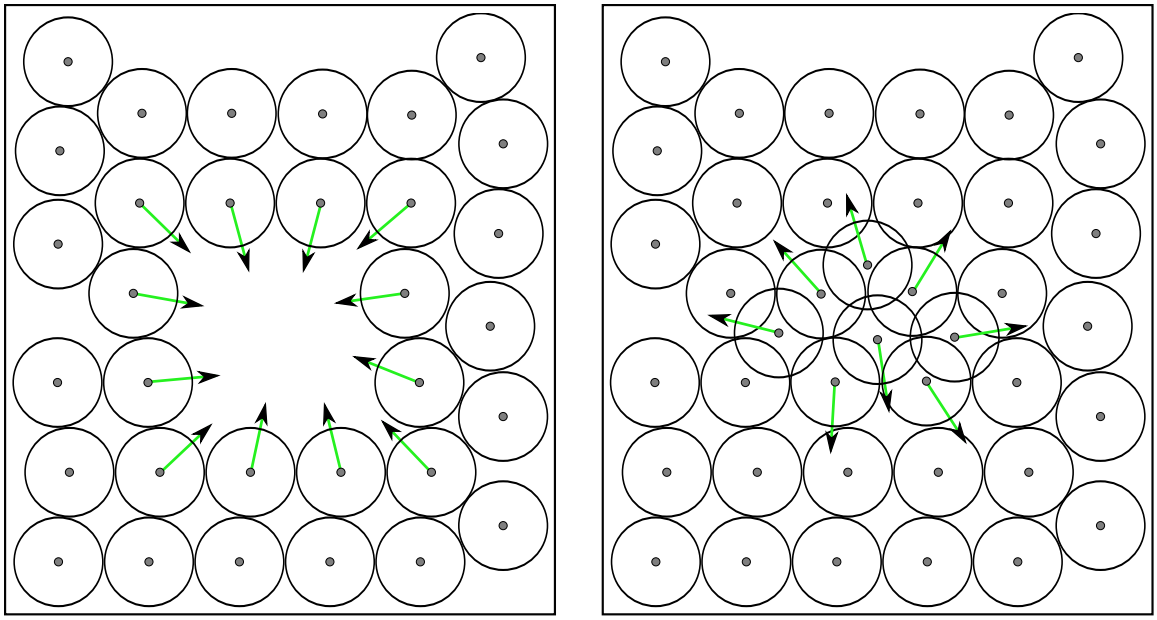
$$f_i^{press} = -\frac{1}{\rho_i}\sum_{j\neq i} p_j \frac{m_j}{\rho_j}\nabla W(\mathbf{r}_{ij}, h). \tag{4.12}$$

Při vzájemném působení dvou částic podle (4.12), nedochází k symetrickému působení sil. Existuje více variant, jak tento problém řešit. Müller [23] navrhuje velmi jednoduchou symetrizaci založenou na aritmetickém průměru

$$\mathbf{f}_i^{press} = -\frac{1}{\rho_i}\sum_{j\neq i} m_j \frac{p_i + p_j}{2\rho_j}\nabla W(\mathbf{r}_{ij}, h). \tag{4.13}$$

Další, složitější symetrizace ukazuje Dalrymple [8]

$$\begin{aligned}\mathbf{f}_i^{press} &= -\sum_{j\neq i} 2m_j \frac{\sqrt{p_i p_j}}{\rho_i \rho_j}\nabla W(\mathbf{r}_{ij}, h),\\ \mathbf{f}_i^{press} &= -\sum_{j\neq i} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right)\nabla W(\mathbf{r}_{ij}, h).\end{aligned} \tag{4.14}$$



Obrázek 4.1: Síly způsobené tlakem v důsledku nedostatku a přebytku částic v oblasti podle výrazu $f_i = -\nabla p$

4.2.2 Viskozita

Viskozita je mírou schopnosti kapaliny odolávat namáhání. Vliv viskozity je patrný v rovnici (2.7). Aplikací metody SPH na výraz $\mu\nabla^2\mathbf{v}$ získáme vztah

$$f_i^{visc} = \frac{\mu}{\rho_i}\sum_{j\neq i} m_j \frac{\mathbf{v}_j}{\rho_j}\nabla^2 W(\mathbf{r}_{ij}, h). \tag{4.15}$$

Síly popsané tímto vztahem jsou ovšem opět nesymetrické. Protože nás ovšem nezajímá absolutní rychlost částice, ale pouze její relativní rychlost vůči referenční částici, existuje velmi přirozený způsob, jak získat symetrizaci působících viskózních sil:

$$f_i^{visc} = \frac{\mu}{\rho_i}\sum_{j\neq i} m_j \frac{\mathbf{v}_i - \mathbf{v}_j}{\rho_j}\nabla^2 W(\mathbf{r}_{ij}, h). \tag{4.16}$$

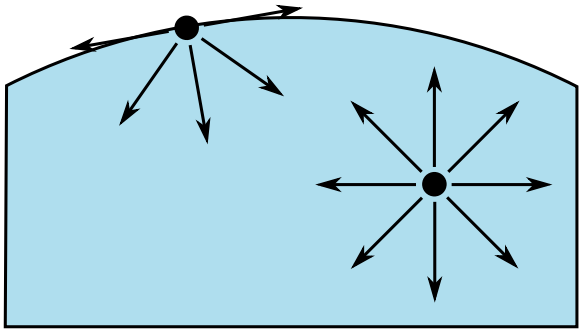
Částice je tedy zrychlena ve směru relativní rychlosti svého okolí.

4.3 Vnější síly

V této práci uvažuji pouze sílu gravitační a sílu vznikající v důsledku povrchového napětí. Dále jsem do této části přidal podkapitolu týkající se hraničních podmínek, které také silově přispívají na jednotlivé částice.

4.3.1 Povrchové napětí

Povrchové napětí je způsobeno přitažlivými silami mezi molekulami kapaliny. Tyto síly, jak je znázorněno na obrázku 4.2, jsou v blízkosti povrchu v nerovnováze.



Obrázek 4.2: Vznik povrchového napětí v důsledku nevyvážených mezimolekulárních sil

Důsledkem povrchového napětí je snaha kapalin zaujmout tvar s nejmenším poměrem povrchu k objemu, tedy tvar s nejmenší křivostí povrchu. V prostředí bez gravitace zaujímají kapaliny tvar koule. Čím je povrch kapaliny křivější, tím je povrchové napětí větší. Povrchové napětí není obsaženo v rovnici (4.1), proto je modelováno jako externí síla.

K modelování povrchového napětí nám poslouží veličina zvaná obarvení c. Obarvení je definováno takto:

$$c(\mathbf{x}) = \begin{cases} 1, & \exists p_i; \mathbf{x}_i = \mathbf{x} \\ 0, & jinak \end{cases}, \tag{4.17}$$

tedy 1 na pozici částice a 0 jinde. Gradient tohoto pole je normálou k povrchu $\mathbf{n} = \nabla c$. Normálu získáme aplikací SPH:

$$\mathbf{n}_i = \sum_{j=1}^{N} m_j \frac{1}{\rho_j} \nabla W(\mathbf{r}_{ij}, h), \tag{4.18}$$

Tato normála má nenulovou velikost pouze v blízkosti povrchu. Výsledný vztah pro křivost povrchu je:

$$\kappa = \frac{-\nabla^2 c_s}{|n|}. \tag{4.19}$$

Pro výpočet κ je tedy možné použít metodu SPH:

$$\nabla^2 c_i = \sum_{j=1}^{N} m_j \frac{1}{\rho_j} \nabla^2 W(\mathbf{r}_{ij}, h). \tag{4.20}$$

Síla působící na každou částici je rovna

$$f_i^{surf} = -\sigma \nabla^2 c_i \frac{\mathbf{n}_i}{|\mathbf{n}_i|}. \tag{4.21}$$

Vzhledem k tomu, že absolutní velikost normály $|\mathbf{n}_i|$ se nachází ve jmenovateli, je vyhodnocení problematické. Pokud je $\mathbf{n}_i$ malé, jedná se o částici daleko od povrchu a význam $\mathbf{f}_i^{surf}$ je zanedbatelný. Proto se tato síla vyhodnocuje pouze pro částice s normálou, jejichž normála má velikost překračující určitý práh.

4.3.2 Gravitace

Gravitační síla působí shodně na všechny částice, není závislá na pozici, okolí ani dalších vlivech. Není třeba použít SPH ani jinou speciální metodu a stačí ke zrychlení a_i na každou částici přičíst gravitační zrychlení g.

4.3.3 Hraniční podmínky

Existuje několik způsobů, jak se vypořádat s hraničními podmínkami.

Perfektní odraz

Jedná se o nejjednodušší variantu, kdy každá částice při kontaktu s překážkou projde perfektním odrazem. Odraz se řídí zákonem dopadu a odrazu. Toto řešení je poměrně naivní a ne příliš přesné, protože kolize ovlivňují jednotlivé částice a nemají vliv přímo na řešení SPH.

Odpudivé síly

Tento způsob řešení přidává na hranice odpudivé síly. Müller [23] implementoval síly, které částice tlačí pryč od překážek ve směru normály povrchu. Velikost odpudivé síly je úměrná hloubce průniku do objektu.

Harada et al. [12] implementovali kolize pomocí úpravy hustoty a potažmo tlaku částic poblíž hranic s pevnými objekty. Takto navýšený tlak zamezí průniku částic do překážky.

Virtuální částice

Toto řešení použili Takeda et al. [31] a Morris [22]. Okolo hranic do vzdálenosti rovné vyhlazovací délce přidali virtuální (také stínové) částice, které mají pevnou pozici v prostoru. Toto řešení je dobré, protože přímo ovlivňuje řešení metodou SPH. Nevýhodou u členitých překážek je, že řešení je ovlivněno i konkrétním rozmístěním stínových částic. Jejich vhodné generování u členitých překážek může být poměrně složitý problém.

Deficit sumy jádra

Hranice v SPH siulacích jsou problematické zejména proto, že vzniká deficit při sumě jádra. Za překážkou se nenacházejí žádné částice a proto hustota vypočtena metodou SPH je menší než u částic uvnitř kapaliny.

4.4 Jádra pro SPH

V této části uvedu seznam podmínek, které by měly jádra splňovat a uvedu přesnou specifikaci specializovaných jader navržených Müllerem [23]. Na závěr zmíním obecná jádra používaná v SPH simulacích.

4.4.1 Vlastnosti jádra

Jádro použité v metodě SPH musí pro svou dobrou funkci splňovat řadu podmínek. Některé z nich jsem už zmínil v kapitole 3, zde je pro úplnost uvedu znovu.

1. **Kompaktnost jádra**

   Jádro musí splňovat následující vztah, který zaručuje reprodukci konstant:

$$\int_v W(\mathbf{x}, h)dv = 1. \tag{4.22}$$

2. **Reprodukující jádro**

   Pro velikost vyhlazovací délky blížící se nule musí jádro odpovídat Diracově funkci delta:

$$\lim_{h\to 0} W(\mathbf{x}, h) = \delta(\mathbf{x}). \tag{4.23}$$

3. **Kladnost jádra**

   Jádro musí být na celém svém definičním oboru kladné, tedy:

$$W(\mathbf{x}, h) >= 0 \;\text{ pro }\; \forall \mathbf{x} \in D(f) \tag{4.24}$$

4. **Sudost jádra**

   Od jádra očekáváme izotropní chování, proto funkce popisující jádro musí být sudá.

$$W(\mathbf{x}, h) = W(-\mathbf{x}, h) \tag{4.25}$$

Všechny tyto podmínky splňuje gaussovské jádro:

$$W_{gauss}(\mathbf{x}, h) = \frac{1}{(2\pi h^2)^{\frac{3}{2}}} e^{-\frac{|x|^2}{2h^2}}, \quad h > 0, \tag{4.26}$$

Toto jádro je vyhovující pro použití v teoretických aplikacích. Pro použití v počítačové simulaci ovšem není vhodné, protože jádro nemá kompaktní definiční obor a musí se připočítávat i příspěvky ze vzdálených částic. Navíc je pro výpočet jádra nutné vyhodnotit exponenciální funkci. Proto se na jádro v praxi klade jedna z dvou následujících alternativních podmínek:

$$\begin{aligned} W(\mathbf{x}, h) &= 0, \quad |\mathbf{x}| > h \\ W(\mathbf{x}, h) &= 0, \quad |\mathbf{x}| > 2h \end{aligned} \tag{4.27}$$

4.4.2 Specializovaná jádra

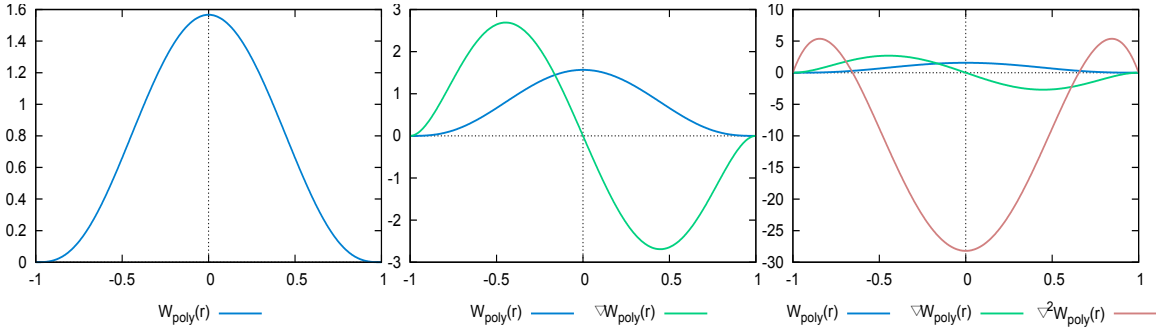
Ve své práci jsem použil specializovaná jádra navržená Müllerem [23]. Následuje jejich podrobný popis.

Standardní jádro

Standardní jádro (obrázek 4.3) je použito všude tam, kde tomu nebrání nějaký specifický důvod. Není třeba vyhodnocovat žádnou speciální funkci ani odmocninu, proto je výpočet jádra rychlý. Rovnice vypadají takto:

$$W_{poly}(\mathbf{x}, h) = \frac{315}{64\pi h^9} \begin{cases} \left(h^2 - |\mathbf{x}|^2\right)^3 & , \quad 0 \le |\mathbf{x}| \le h \\ 0 & , \quad jinak \end{cases}, \tag{4.28}$$

$$\nabla W_{poly}(\mathbf{x}, h) = -\frac{945}{32\pi h^9} \left(h^2 - |\mathbf{x}|^2\right)^2 \mathbf{x} \;\; , \;\; 0 \le |\mathbf{x}| \le h, \tag{4.29}$$

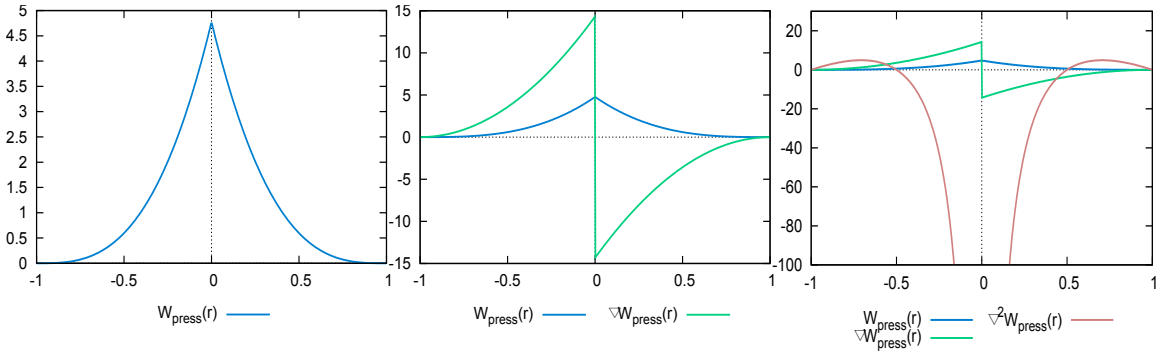


Obrázek 4.3: Standardní jádro, průmět do jedné dimenze, pro $h = 1$. Narozdíl od gaussovského jádra má toto jádro omezený support

$$\nabla^2 W_{poly}(\mathbf{x}, h) = -\frac{945}{32\pi h^9}\left(h^2 - |\mathbf{x}|^2\right)\left(3h^2 - 7\,|\mathbf{x}|^2\right) \quad , \quad 0 \leq |\mathbf{x}| \leq . \tag{4.30}$$

Jak je vidět, z výše uvedených rovnic pro popis standardního jádra (4.28), gradientu standardního jádra (4.29) a laplaciánu standardního jádra (4.30), část rovnic, která se nachází ve zlomku v levé části lze předpočítat, protože není závislá na $\mathbf{x}$.

Jádro pro výpočet tlaku



Obrázek 4.4: Jádro pro výpočet tlaku, průmět do jedné dimenze, pro $h = 1$.

Na obrázku 4.3, znázorňujícím průběh standardního jádra, je vidět, že gradient tohoto jádra není vhodný k použití v metodě SPH, protože platí, že $\lim_{\mathbf{x}\to 0} = 0$. Při výpočtu gradientu tlakového pole je tedy třeba použít jádro jiné, jehož gradient splňuje požadavky na jádro. Takové jádro je např. uvedené na obrázku 4.4. Toto jádro je popsáno následujícími rovnicemi:

$$W_{press}(\mathbf{x}, h) = \frac{15}{\pi h^6}\begin{cases} (h - |\mathbf{x}|)^3 & , \quad 0 \leq |\mathbf{x}| \leq h \\ 0 & , \quad jinak \end{cases} \tag{4.31}$$

$$\nabla W_{press}(\mathbf{x}, h) = -\frac{45}{\pi h^6}\frac{\mathbf{x}}{|\mathbf{x}|}(h - |\mathbf{x}|)^2\,, \tag{4.32}$$

$$\lim_{x\to 0^-} \nabla W_{press}(x, h) = \frac{45}{\pi h^6} \quad , \quad \lim_{x\to 0^+} \nabla W_{press}(x, h) = -\frac{45}{\pi h^6},$$

$$\begin{aligned} \nabla^2 W_{press}(\mathbf{x},h) &= -\frac{90}{\pi h^6}\frac{1}{|\mathbf{x}|}\left(h-|\mathbf{x}|\right)\left(h-2\,|\mathbf{x}|\right) \quad , \quad 0 \le |\mathbf{x}| \le h \\ \lim_{x\to 0} \nabla W_{press}(x,h) &= -\infty, \end{aligned} \tag{4.33}$$

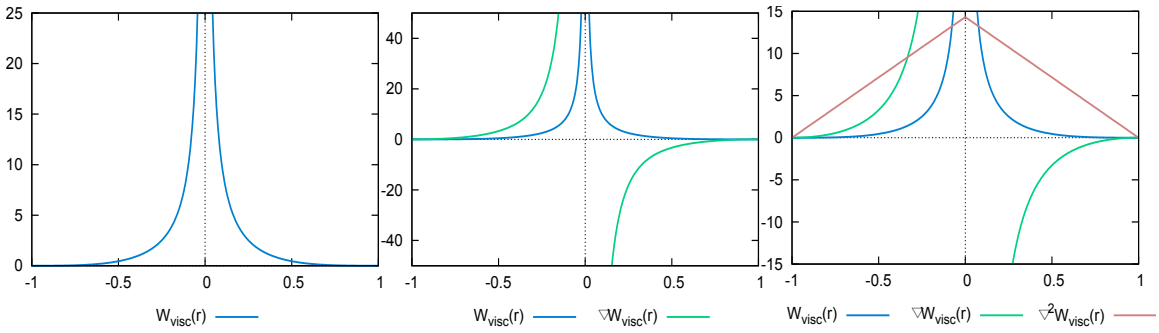
Jádro pro výpočet viskozity

Pro výpočet viskozity potřebujeme použít laplacián jádra, nemůžeme tedy použít žádné z předcházejících jader, protože obě mají na části svého definičního oboru záporný laplacián. Je možné použít jádro, které je na obrázku 4.5. Jádro je popsáno rovnicemi:

$$\begin{aligned} W_{visc}(\mathbf{x},h) &= \frac{15}{2\pi h^3}\begin{cases} \frac{|\mathbf{x}|^3}{2h^3}+\frac{|\mathbf{x}|^2}{h^2}+\frac{h}{2|\mathbf{x}|}-1, & 0 \le |\mathbf{x}| \le h \\ 0, & jinak \end{cases}, \\ \lim_{x\to 0} W_{visc}(x,h) &= \infty, \end{aligned} \tag{4.34}$$

$$\begin{aligned} \nabla W_{visc}(\mathbf{x},h) &= \frac{15}{2\pi h^3}\mathbf{x}-\frac{3\,|\mathbf{x}|}{2h^3}+\frac{2}{h^2}+-\frac{h}{2\,|\mathbf{x}|^3}, \quad 0 \le |\mathbf{x}| \le h \\ \lim_{x\to 0^-} W_{visc}(x,h) &= +\infty, \quad \lim_{x\to 0^+} W_{visc}(x,h) = -\infty, \end{aligned} \tag{4.35}$$

$$\nabla^2 W_{visc}(\mathbf{x},h) = \frac{45}{\pi h^6}\left(h-|\mathbf{x}|\right). \tag{4.36}$$



Obrázek 4.5: Jádro pro výpočet viskozity, průmět do jedné dimenze, pro $h = 1$.

4.4.3 Obecná jádra

Pro úplnost uvedu seznam obecných jader, různých autorů, která byla použita v simulacích metodou SPH.

- **Gaussovské jádro** – Lze obecně použít, je ovšem pomalé.
- **Piecewise Cubic Spline** – Velmi hojně používané jádro, druhý derivát není hladký, nevhodné pro výpočet laplaciánu (Monaghan [21]).
- **Piecewise Quintic Spline** – Používáno u toků s nízkým Reynoldsovým číslem, velmi výpočetně náročné (Morris et al. [22]).

- **Quadratic** – Dobře zabraňuje shlukování částic (Crespo [7]).
- **Wendland** – Kompromis mezi rychlostí a přesností (Wendland [35] a Crespo [7]).
- **Quartic** – Podobné jako Piecewise Cubic Spline, ale není třeba jej vyhodnocovat po částech (Liu et al.[16]).

Kapitola 5

Návrh řešení a implementace

Tato kapitola obsahuje návrh řešení a popis zajímavých částí implementace algoritmu SPH. Kapitola obsahuje analýzu paměťové náročnosti algoritmu. Rozbor vlivu parametrů simulace a jiné.

5.1 Použité technologie

Program jsem se rozhodl implementovat v jazyku C++ (CPU) a v jazyce C for CUDA (GPU). Aplikace je navržená, aby využívala grafický procesor k paralelním výpočtům. Rozhodnul jsem se využít technologii CUDA firmy NVIDIA, proto lze program spustit pouze v systémech s CUDA-enabled grafickým procesorem firmy NVIDIA (GeForce, Tesla…). Program vyžaduje CUDA capability 1.2. Vizualizace simulace je realizována za použití knihovny OpenGL. Aplikace je vytvořena pro operační systémy Windows.

5.2 Návrh řešení

Konečným produktem této diplomové práce by měl být počítačový program simulující viskózní kapaliny a software pro vizualizaci výstupu simulací. Konkrétním příkladem simulovaných systému (scénářů simulace) je např. prolomení přehrady nebo simulace vln.

5.2.1 Algoritmus simulace

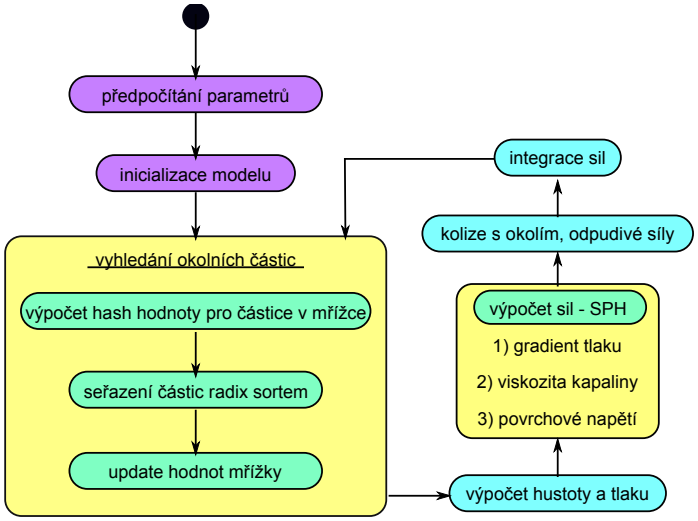
Algoritmus simulace se sestáva z několika základních kroků, které jsou zobrazeny na schématu 5.1. Jednotlivé kroky algoritmu stručně popíšu:

1. **Předpočítání parametrů**

   Jedná se zejména o předpočítání neměnných částí jader, viz sekce 4.4. Dále pak výpočet vyhlazovací délky h podle vztahu

   $$h = \sqrt[3]{23\frac{3m}{4\pi\rho_0}}, \tag{5.1}$$

   kde 23 je konstanta zvolená po vzoru Nováka [25] tak, aby odpovídala počtu částic v okolí h pro kapalinu v klidu s hustotou rovnou ρ_0.



Obrázek 5.1: Diagram průběhu simulace

2. **Inicializace modelu**

   V této části simulace se vygeneruje startovní pozice všech částic v systému, jejich počáteční rychlost. Rozložení pozic částic a jejich rychlostí pak závisí na konkrétní aplikaci. Obecně je třeba částice rozmístit tak, aby vzdálenost sousedních částic alespoň přibližně odpovídala klidové vzdálenosti částic. V případě malé vzdálenosti mezi částicemi by po spuštění simulace v důsledku velkého tlaku došlo k „rozletění“ částic do okolí.

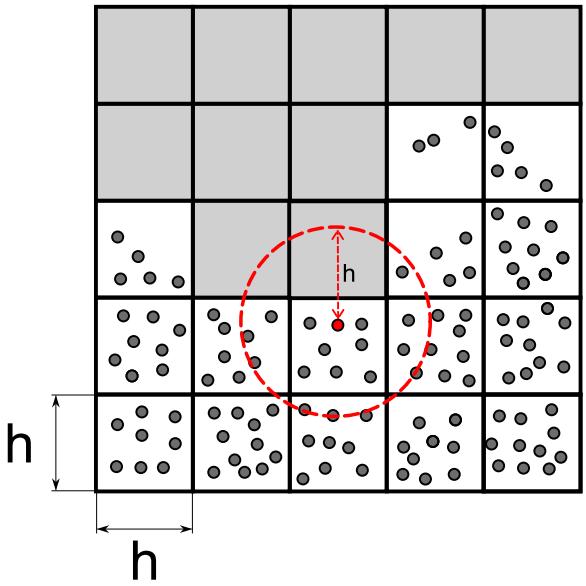
3. **Vyhledání okolních částic**

   Vyhledání okolních částic je realizováno pomocí uniformní mřížky. Tato mřížka je sestrojena tak, že velikost hrany každé buňky je rovna vyhlazovací délce h. Díky použití omezených jader pro metodu SPH, je příspěvek částic vzdálenějších než h (vyhlazovací délka) nulový, což umožňuje omezit prohledávání na 27 buněk v nejbližším okolí částice (ve dvojrozměrném případě je to znázorněno na obrázku 5.2). Hlavním důvodem k použití mřížky je snaha o snížení časové složitosti vyhledávání, která je bez jakékoliv optimalizace rovná $O(n^2)$. Vyhledávání lze dále rozdělit na následující tři kroky. Tyto kroky jsou podrobněji popsány v kapitole 5.3.3.

   (a) Výpočet hash hodnoty pro částice v mřížce

   (b) Seřazení částic

   (c) Aktualizace hodnot mřížky



Obrázek 5.2: Znázornění pravidelné mřížky pro urychlení vyhledávání částic (dvojrozměrný případ)

4. **Výpočet hustoty a tlaku**

   Pro každou částici se metodou SPH podle vztahu (4.3) vypočte hustota asociovaná s danou částicí. Poté se vypočte tlak pro každou částici ze stavové rovnice plynů (4.8).

5. **Výpočet sil**

   Z důvodu efektivity by bylo nejlepší provést všechny výpočty metodou SPH v jednom průchodu mřížkou. To ovšem bohužel není možné, protože ke všem výpočtům, kromě výpočtu hustoty, je potřeba znát hustotu příslušející každé částici. Proto je výpočet sil působících na částice umístěn do samostatné fáze simulace. Síla na částici je způsobena těmito fyzikálními jevy.

   (a) **Gradient tlaku**
   Výpočet probíhá podle vztahu (4.13)

   (b) **Viskozita kapaliny**
   Výpočet probíhá podle vztahu (4.16).

   (c) **Povrchové napětí**
   Výpočet probíhá podle vztahu (4.21).

6. **Kolize s okolím, odpudivé síly**

   V tomto kroku jsou spočítány kolize s okolím. V tomto jednoduchém případě se jedná zejména o kolize s obklopující nádobou a jednoduchými tělesy uvnitř nádoby. Kolize s okolím jsem implementoval jako pružné podle vzoru [1], podrobněji viz sekce o implementaci.

7. **Numerická integrace**

   Po sečtení sil působících na částici vypočteme zrychlení na částici. Rychlost a pozici částice v čase t získáme numerickou integrací metodou Leap-frog.

$$\mathbf{r}_{(t+1)} = \mathbf{r}_{(t)} + \mathbf{v}_{\left(t+\frac{1}{2}\right)}\delta t \tag{5.2}$$

$$\mathbf{v}_{\left(t+\frac{1}{2}\right)} = \mathbf{v}_{\left(t-\frac{1}{2}\right)}\mathbf{a}_{(t)}\delta t \tag{5.3}$$

5.2.2 Vizualizace

Aplikace pro vizualizaci slouží k znázornění stavových veličin, jmenovitě tlaku, rychlosti a působící síly na jednotlivé částice. K vizualizaci je použito `OpenGL`. Částice jsou vizualizována pomocí vertex shaderu, který pracuje na datech, zapisovaných přímo z grafické karty do struktur typu `Vertex Buffer Object`.

5.3 Implementace

Samotná implementace algoritmu SPH není příliš složitá věc. Velkou výzvou je ovšem vytvořit implementaci efektivní. Při tvorbě aplikace měla rychlost prioritu před přesností simulace, proto jsem použil nejméně výpočetně náročná jádra navržená Müllerem [23] a hraniční podmínky bez využití stínových částic, viz kapitola 4. V této části popíšu zajímavější části implementace, které jsou klíčové pro efektivitu algoritmu.

5.3.1 Využití paměti

V globální paměti je pro každou částici v časovém kroku i třeba udržovat následující data :

- pozici v čase i, jedná se o vstup i výstup simulačního kroku.
- rychlost v čase $i - \frac{1}{2}$ (kvůli použití integrační metody leap-frog)
- rychlost v čase i, využívá se jako vstup do simulačních výpočtů a získá se ze vztahu $v_i = \frac{1}{2}(v_{i-1/2} + v_{i+1/2})$
- barvu částice, využívá se jako výstup pro renderování, barva záleží na zvoleném barevném schéma a barevném zdroji.

Dále je třeba uchovávat tato seřazená data:

- pozici v čase i, díky seřazení seznamu je urychleno vyhledávání a seřazený seznam pozic se využívá při všech výpočtech metodou SPH
- rychlost v čase $i - \frac{1}{2}$, využívá se při numerické integraci
- rychlost v čase i, využívá se např, při výpočtu viskózních sil a kolizí
- hustotu v místě částice, využívá se ve všech SPH výpočtech (vychází z definice SPH)
- tlak v místě částice pro výpočet tlakové síly a určení barvy částice

- barvu částice pro renderování, barva se určuje ze seřazených bufferů a pak se pomocí původního indexu zkopíruje do neseřazeného seznamu
- sílu způsobenou vnitřními silami — v dalším kroku se na ni aplikují síly vnější

Celkově lze tedy shrnout nároky jedné částice na globální paměť do tabulky 5.1

Atribut	Datový typ	velikost v Bytech
Position	float4	16
Velocity	float4	16
Veleval(leap-frog)	float4	16
Color	float4	16
PositionSorted	float4	16
VelocitySorted	float4	16
VelevalSorted	float4	16
ColorSorted	float4	16
DensitySorted	float	4
PressureSorted	float	4
ForceSorted	float4	16
Celkem		152

Tabulka 5.1: Paměťové nároky jedné částice v systému

Z důvodů řazení částic a propojení původních a seřazených částic je třeba navíc uchovávat v paměti data, která v podstatě reprezentují uniformní mřížku (podrobněji v části 5.3.3):

- hash hodnoty částic (klíče pro třídící algoritmus)
- seřazené indexy částic pro nalezení původních částic v neseřazených seznamech
- index začátků buněk do seřazeného pole pro rychlé vyhledávání sousedních částic
- index konců buněk do seřazeného pole

Tato data lze shrnout do následující tabulky 5.2.

Buffer	Počet prvků	Datový typ	Počet bytů/prvek
SortHashes	= počet částic	uint	4
SortIndices	= počet částic	uint	4
CellStartIndices	= počet buněk mřížky	uint	4
CellEndIndices	= počet buněk mřížky	uint	4

Tabulka 5.2: Data pro mřížku, řazení a vyhledávání

Paměťové nároky na mřížku a struktury pro vyhledávání lze vyjádřit vztahem:

$$8 * n_{particles} + 8 * n_{cells} + 8.375 * n_{particles} B, \tag{5.4}$$

kde $8.375 * n_{particles}$ je, jak uvádí Krog [14], paměť alokovaná použitým algoritmem radixsort. Zřejmě by bylo vhodnější, kdyby se celková spotřeba paměti těchto struktur dala vyjádřit

Počet částic	32768	131072	524288	2097152
Poče buněk mřížky	32768	132651	512000	2048282

Tabulka 5.3: Závislost počtu buněk mřížky na počtu částic

pouze v závislosti na počtu částic v systému. Vztah mezi počtem částic a počtem buněk mřížky je v tabulce 5.3. Z tabulky 5.3 je vidět, že počet částic zhruba odpovídá počtu buněk uniformní mřížky. Jinými slovy, když zmenšujeme částice (změnšuje se jejich hmotnost), zjemňujeme detail simulace a je potřeba také zjemnit uniformní mřížku. Proto můžeme vztah 5.4 přepsat na tvar:

$$16 * n_{particles} + 8.375 * n_{particles} = 24.375 * n_{particles} B, \tag{5.5}$$

Celková spotřeba globální paměti je pak dohromady s pamětí pro data částice rovna:

$$(152 + 24.375) * n_{particles} = 176.375 * n_{particles} B. \tag{5.6}$$

Z výše uvedeného vztahu jasně vyplývá omezení počtu částic vzhledem k dostupné paměti na grafické kartě. Například pro kartu GeForce GT 240M, na které jsem projekt vyvíjel, s pamětí 1010240kB to znamená, že je teoreticky možné simulovat systém s 5865262 částicemi.

5.3.2 Předpočítání parametrů

Vzhledem k rychlosti simulace je vhodné předpočítat části jader, které jsou konstatní, tedy nezávisí na konkrétní vzdálenosti dvou částic. Vzhledem k faktu, že všechny částice kapaliny mají stejnou, neměnnou hmotnost, je možné ji také vytknout z SPH sumy. Obecně tedy výpočty metodou SPH nabývají tvaru:

$$A_i = m \cdot W_{const\_part} \sum_{j \neq i} \frac{A_j}{\rho_j} W_{variable\_part} \tag{5.7}$$

Po aplikaci na konkrétní SPH výpočty získáme tyto vztahy pro výpočet hustoty ρ, tlakové síly f_i^{press}, síly způsobené viskozitou f_i^{visc}, normály povrchu $\mathbf{n}$ a křivosti povrchu K:

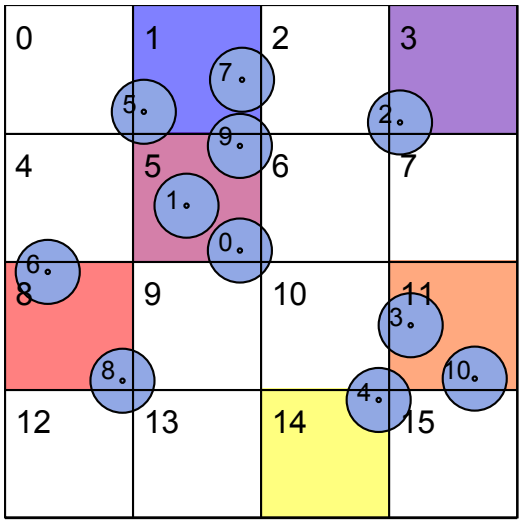
$$\rho_i = m W_{const}^{poly6} \sum_{j \neq i} W_{var}^{poly6} \tag{5.8}$$

$$\mathbf{f}_i^{press} = -\frac{m}{2} \frac{1}{\rho_i} \nabla W_{const}^{press} \sum_{j \neq i} \frac{p_i + p_j}{\rho_j} \nabla W_{var}^{press} \tag{5.9}$$

$$\mathbf{f}_i^{visc} = \mu m \frac{1}{\rho_i} \nabla^2 W_{const}^{visc} \sum_{j \neq i} \frac{\mathbf{v}_j - \mathbf{v}_i}{\rho_j} \nabla^2 W_{var}^{visc} \tag{5.10}$$

$$\mathbf{n}_i = m \nabla W_{const}^{poly6} \sum_{j \neq i} \frac{1}{\rho_j} \nabla W_{var}^{poly6} \tag{5.11}$$

$$\nabla^2 c_i = m \nabla^2 W_{const}^{poly6} \sum_{j \neq i} \frac{1}{\rho_j} \nabla^2 W_{var}^{poly6} \tag{5.12}$$



index	neseřazený seznam (id buňky, id částice)	seznam seřazený podle id buňky	(začátek buňky, konec buňky)
0	(5, 0)	(1, 5)	
1	(5, 1)	(1, 7)	(0, 1)
2	(3, 2)	(3, 2)	
3	(11, 3)	(5, 0)	(2, 2)
4	(14, 4)	(5, 1)	
5	(1, 5)	(5, 9)	(3, 5)
6	(8, 6)	(8, 6)	
7	(1, 7)	(8, 8)	
8	(8, 8)	(11, 3)	(6, 7)
9	(5, 9)	(11, 10)	
10	(11, 10)	(14, 4)	
11			(8, 9)
12			
13			
14			(10,10)
15			

Obrázek 5.3: Obrázek s tabulkou znázorňující třídění částic a tvorbu uniformní mřížky. Levý sloupec tabulky představuje indexy do všech seznamů(polí). Druhý sloupec zobrazuje původní, nesetříděný seznam částic a jejich příslušnost do buněk mřížky. Ve třetím sloupci jsou tyto částice seřazeny právě podle čísel buňky. V posledním sloupci obsahuje i-tý řádek dvojici indexů do seřazeného seznamu částic, ukazujících na první respektive poslední částici patřící do i-té buňky.

Definice částí jader z rovnic 5.8 až 5.12 jsou následující:

$$W_{const}^{poly6} = \frac{315}{64\pi h^9} \tag{5.13}$$

$$W_{var}^{poly6}(\mathbf{r}, |\mathbf{r}|^2, h) = (h^2 - |\mathbf{r}|^2)^3 \tag{5.14}$$

$$\nabla W_{const}^{press} = \nabla^2 W_{const}^{visc} = \frac{45}{\pi h^6} \tag{5.15}$$

$$\nabla W_{var}^{press}(\mathbf{r}, |\mathbf{r}|, h) = \frac{\mathbf{r}}{|\mathbf{r}|}(h - |\mathbf{r}|)^2 \tag{5.16}$$

$$\nabla^2 W_{var}^{visc}(\mathbf{r}, |\mathbf{r}|, h) = (h - |\mathbf{r}|) \tag{5.17}$$

$$\nabla W_{const}^{poly6} = \nabla^2 W_{const}^{poly6} = -\frac{945}{32\pi h^9} \tag{5.18}$$

$$\nabla W_{var}^{poly6}(\mathbf{r}, |\mathbf{r}|^2, h) = (h^2 - |\mathbf{r}|^2)^2 \mathbf{r} \tag{5.19}$$

$$\nabla^2 W_{var}^{poly6}(\mathbf{r}, |\mathbf{r}|^2, h) = (h^2 - |\mathbf{r}|^2)(3h^2 - 7|\mathbf{r}|^2) \tag{5.20}$$

5.3.3 Vyhledávání okolních částic

Objem simulované kapaliny je rozdělen do uniformní 3D mřížky. Každá buňka této mřížky je reprezentována třemi prostorovými souřadnicemi x, y a z. Velikost buněk mřížky je zvolena tak, aby byla rovna vyhlazovací délce h. To umožňuje omezit prohledávání sousedních částic na okolních 27 buněk, tedy na buňku, ve které se nachází aktuální částice a všechny buňky přilehlé (viz obrázek 5.2).

Příslušnost částice do buňky (záleží na pozici středu částice) se vypočítá podle vztahu:

$$\mathbf{cell\_index}_j = floor\left(\frac{\mathbf{pos}_j - \mathbf{grid\_min}}{size_{cell}}\right), \tag{5.21}$$

kde $\mathbf{cell\_index}_j$ je výsledný index, operace $floor$ provádí zaokrouhlení směrem dolů, $\mathbf{pos}_j$ je souřadnice dané částice, **grid_min** je minimální souřadnice náležící do mřížky a $size_{cell}$ je velikost buňky. Tímto postupem získáme indexy buňky odpovídající souřadnicím x, y a z. Částice je třeba seřadit tak, aby ty, které náleží do stejné buňky, následovaly po sobě. Každé buňce přísluší hash používaný pro řazení, získaný ze vztahu:

$$hash = y_{max}x_{max}z \ + x_{max}y + x. \tag{5.22}$$

Podle této hash hodnoty se seřadí (pomocí algoritmu radix sort z knihovny Cudpp) indexy částic uložené ve struktuře reprezentující mřížku. Index na $i$té pozici v seřazeném poli indexů ukazuje na pozici částice v neseřazeném seznamu. Za pomocí seřazených indexů se rychle seřadí ostatní seznamy obsahující data částic. V dalším kroku se prohledáním seřazených seznamů částic a indexů naleznou indexy začátků a konců jednotlivých buněk. Tyto indexy se využijí k rychlému nalezení všech částic určité buňky. Algoritmus je použit v příkladu CUDA SDK Simona Greena [11].

Pseudokód pro procházení částic v sousedních buňkách

```
cell_index = CalculateGridCell(position_i)
for x = -1; x <= 1; x++
  for y = -1; y <= 1; y++
    for z = -1; z~<= 1; z++
      offset =  make_float3(x, y, z)
      hash = CalculateGridHash(cell_index + offset)
      for i = cell_start_indices[hash]; i <= cell_end_indices[hash]; i++
        CalculateSPH(sorted_particles[i])
```

5.3.4 Kolize s okolím

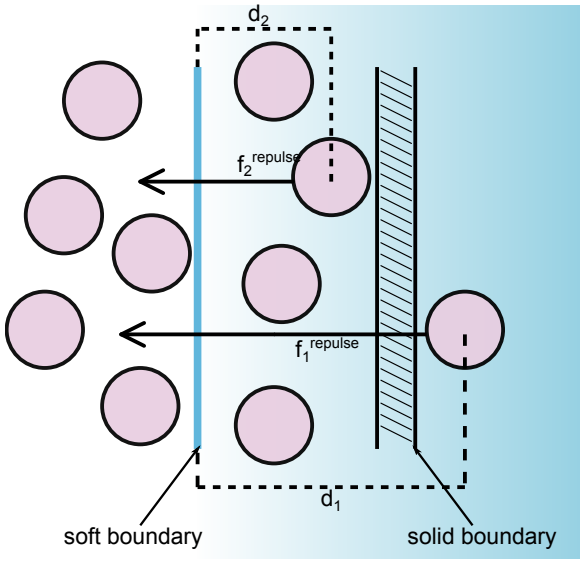
Kolize s okolím zahrnují kolize s kontejnerem obsahujícím nádobu a s překážkami umístěnými uvnitř kontejneru. Kolize jsou implementovány tak, že částice je odpuzována silou, která je přímo úměrná hloubce průniku do překážky. Síla je vypočtena po vzoru Amady [1] podle vztahu

$$\mathbf{f}_i^{repulse} = \begin{cases} K_s d - ((\mathbf{v}_i \cdot \mathbf{n})\, K_d)\, \mathbf{n} & , \quad d \geq \epsilon \\ 0 & , \quad jinak \end{cases}, \tag{5.23}$$

kde K_s je tvrdost překážky, K_d je tlumící parametr, d je hloubka průniku do překážky, ϵ je přesnost srážky (malé číslo), $\mathbf{v}_i$ je rychlost částice a $\mathbf{n}$ je normála povrchu překážky.

Tato metoda je výhodná v tom, že odpudivá síla překážky z části nahrazuje deficit tlaku na rozhraní s překážkami. Nevýhodou je, že částice pod velkým tlakem pronikají do překážky — překážka není pevná. Tento jev lze částečně omezit zmenšením velikosti simulačního kroku, to ovšem není zhlediska rychlosti simulace žádoucí. Pronikání částic do překážek lze vyřešit „přesunutím“ částice za hranice překážky.

Uvážíme-li však, že částice do překážky pronikají v důsledku velkého tlaku proudící kapaliny, není výše uvedené řešení vyhovující. Částice přesunutá za hranici překážky zvýší v daném místě beztak vysoký tlak, což vede k velkým tlakovým silám, které v případě velkého časového kroku vede k lokálním nestabilitám, tedy vystřelení částice. Tento problém jsem vyřešil tak, že zmenším hloubku průniku — tedy změním pozici částice — na polovinu.



Obrázek 5.4: Znázornění odpudivé síly při kolizi s hranicí kontejneru. d_1 a d_2 jsou hloubky průniku do překážky, ke kterým přísluší jednotlivé odpudivé síly. Z obrázku je zřejmé, že odpudivé síly začínají působit již od „měkké hranice“, která zabraňuje částicím dostat se za skutečnou hranici přepážky. Je vidět, že odpudivá síla je úměrná hloubce průniku.

Kolize se stěnami kontejneru

Detekovat kolize se stěnami kontejneru je triviální problém. Stačí kontrolovat souřadnice částice, jestli nepřekročí meze definované kontejnerem.

Kolize s objekty ve scéně

Z důvodu jednoduchosti jsem implementoval pouze kolize s tělesy tvaru kvádru. V principu by ovšem výpočet kolizí s obecnými tělesy definovanými trojuhelníkovou síti a příslušnými normálami byl možný. Kolize s objektem zřejmě nastává tehdy, když se částice nachází uvnitř objemu tělesa. Při kolizích s objekty ve scéně může nastat několik problémů.

- Částice může z důvodu vysoké rychlosti a velkého časového kroku „proletět“ tenkou překážkou.
- Je potřeba určit stranu, se kterou došlo ke kolizi. To je možné vyřešit spočítáním průsečíku obráceného vektoru rychlosti s tělesem. To je ovšem výpočetně náročné, proto jsem kolize implementoval tak, že částice je vypuzována z tělesa nejkratší cestou, tedy nejbližší stěnou kvádru. To ovšem opět znamená, že částice může tělesem při velkém časovém kroku projít.

5.3.5 Výpočet hustoty a sil

V následujích algoritmech je pro průchod všech částic v okolních buňkách mřížky, tak jak je popsán v části 5.3.3 použit zkrácený zápis.

Hustota

Výpočet hustoty metodou SPH

```
for all particles i:
  density_i = 0
  for all neighboring_cells c:
    for all particles j in cell c:
    vector r = particle_pos_i - particle_pos_j
    scalar r_len = length(r)
    if(rlen <= smoothing_length)
      density_i += W_poly6_var(smoothing_length, r, rlen)
  density_i *= particle_mass * W_poly6_const
```

Výpočet sil

Výpočet sil způsobených gradientem tlaku, sil viskozních a sil způsobených povrchovým napětím probíhá v jednom kroku.

Výpočet sil metodou SPH

```
for all particles i:
  force_i = 0
  vector f_press = 0
  vector f_visc = 0
  vector f_surf_tension = 0
  vector surf_normal = 0
  vector surf_curve = 0

  for all neighboring_cells c:
    for all particles j in cell c:
    vector r = particle_pos_i - particle_pos_j
    scalar r_len = length(r)
    if(rlen <= smoothing_length)
      f_press += ((press_i + press_j)/density_i * density_j) *
                  W_press_grad_var(smoothing_length, r, rlen)
      f_visc += ((vel_i - vel_j)/density_i * density_j) *
                 W_visc_laplace_var(smoothing_length, r, rlen)
      surf_normal += 1/(density_i * density_j) *
                      W_poly6_grad_var(smoothing_length, r, rlen)
      surf_curve += 1/(density_i * density_j) *
                     W_poly6_laplace_var(smoothing_length, r, rlen)

  force_i += f_press * particle_mass * W_press_grad_const
  force_i += f_visc * particle_mass * W_visc_laplace_const

  sufr_normal *= particle_mass * W_poly6_grad_const
  surf_curve *= - particle_mass * W_poly6_laplace_const / length(surf_normal)
  f_surf_tension = surf_tension_coeff * surf_normal * surf_curve
  force_i += f_surf_tension
```

5.3.6 Vliv parametrů simulace

Simulace je řízena sadou parametrů, jejichž význam popíšu v této části. V následujících seznamech budou parametry pojmenovány česky, pokud je možné daný parametr zadat přes konfigurační soubor, bude příslušející pojmenování parametru pro konfigurační soubor uvedeno v hranatých závorkách. Ukázkový konfigurační soubor je v příloze B.

Vlastnosti kapaliny

Jedná se o parametry, které přímo ovlivňují fyzikální vlastnosti simulované kapaliny. Některé z těchto parametrů se pomocí konfiguračního souboru zadávají jako vstup do simulace, jiné se vypočítají přímo z ostatních zadaných parametrů.

- **Počet částic** `[PARTICLES_NUMBER]`
  Počet částic kapaliny určuje detail simulace, s počtem částic rostou paměťové nároky, viz sekce 5.3.1, počet částic také ovlivňuje hmotnost částic, viz sekce 6.1.1 a nepřímo další parametry jako např. vyhlazovací délku. Pro malé počty částic systém ztrácí chování kapalin.

- **Klidová hustota** `[REST_DENSITY]`
  Výraz klidová hustota má význam, protože kapaliny jsou pomocí SPH modelovány jako slabě stlačitelné (stlačitelnost je řízena parametrem plynová konstanta). Klidová hustota má význam při výpočtu tlaku v místě částice pomocí upravené stavové rovnice pro plyny (4.8).

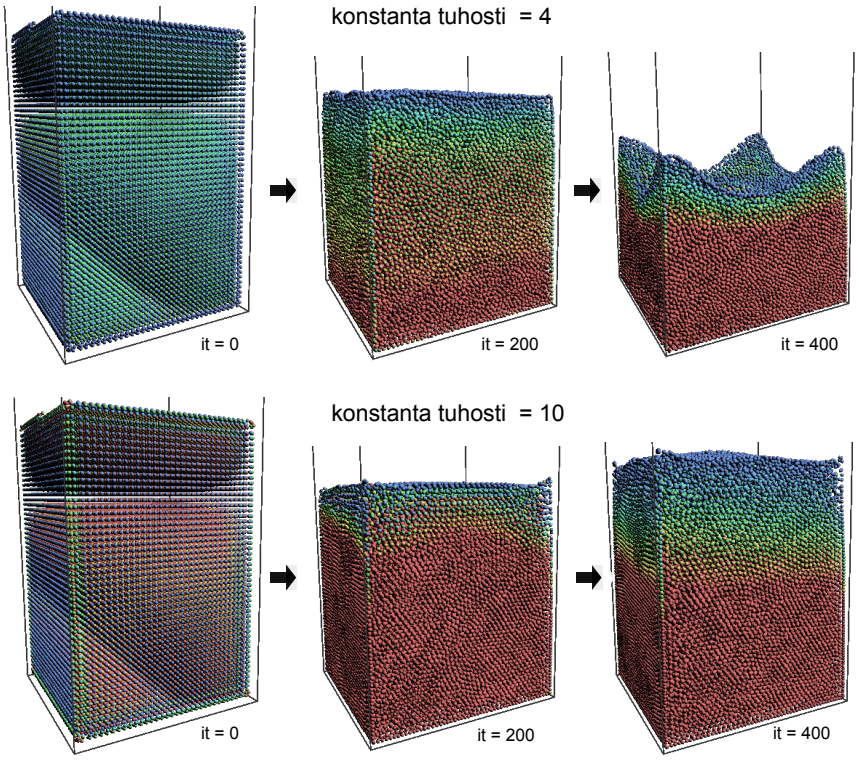
- **Klidový tlak** `[REST_PRESSURE]`
  Klidová hustota figuruje při výpočtu tlaku v místě částice pomocí upravené stavové rovnice pro plyny (4.8).

- **Plynová konstanta** `[IDEAL_GAS_CONSTANT]`
  Plynová konstanta stejně jako dva předešlé parametry figuruje ve stavové rovnici pro výpočet tlaku a zásadním způsobem ovlivňuje stlačitelnost kapaliny viz obrázek 5.5. Volba velikosti plynové konstanty je důležitá z hlediska realističnosti chování kapaliny. Při nízkých hodnotách je kapalina příliš stlačitelná a chová se jako pružina. Při vyšších hodnotách plynové konstanty vznikají v kapalině velké tlaky v jejichž důsledku může dojít k nestabilitám. Aby jim bylo zamezeno, je třeba použít menší časový krok.

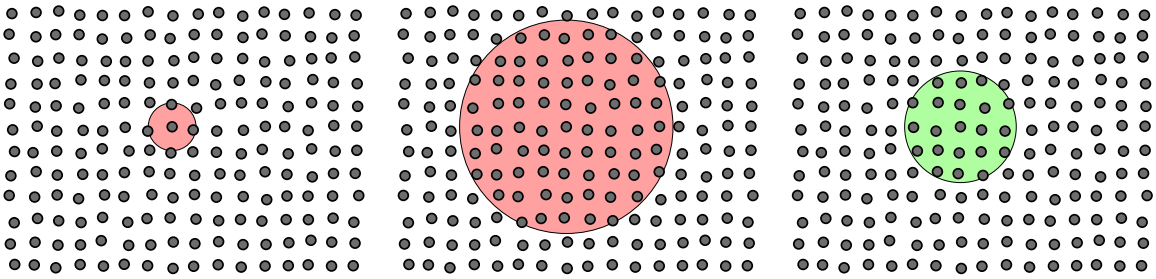
- **Viskozita** `[VISCOSITY]`
  Viskozita je veličina charakterizující vnitřní tření mezi částicemi kapaliny. Kapaliny s větší viskozitu, větší viskozita znamená větší brždění pohybu kapaliny.

- **Hmotnost částic**
  Hmotnost částic závisí na jejich počtu. Čím více částic v systému, tím jsou částice lehčí. Hmotnost částic se vypočte podle vztahu: $m = 128k/n_particles * 0.0002kg$

- **Vyhlazovací délka** Vyhlazovací délka je parametr důležitý z hlediska veškerých SPH výpočtů. Někteří autoři (např. Attwood [2]) používají proměnnou vyhlazovací délku v závislosti na hustotě v místě částice. Pro místa s větší hustotou je vyhlazovací délka kratší. V této práci jsem z důvodu rychlosti simulace použil konstantní vyhlazovací délku, která se vypočítá ze vztahu (5.1). Důležitost volby správné vyhlazovací délky je patrná z obrázku 5.6.



Obrázek 5.5: Kapalina je v počátečním stavu generována do pravidelné mřížky. V horní části je kapalina s menší konstantou tuhosti. Je vidět, že se ustálí později než kapalina s vyšší hodnotou konstanty.



Obrázek 5.6: Vyhlazovací délka by neměla do okolí zahrnout příliš mnoho ani příliš málo částic

- **Klidová vzdálenost částic**
  Jedná se o parametr, který je používán při inicializaci simulace zejména pro rozmístění částic do „klidového“ stavu. Hodnota je rovna polovině vyhlazovací délky.

Parametry scény

Jedná se o parametry ovlivňující měřítko simulace, kolize s hranicemi a neovlivňují přímo fyzikální podstatu simulované kapaliny.

1. **Velikost mřížky** `[GRID_WORLD_SIZE]`
   Pomocí velikosti mřížky lze určit poměrný objem kapaliny v kontejneru. Fyzikálně se sice objem kapaliny nemění (konstantní zůstává jak hmotnost částic, tak jejich počet), ale mění poměr objemu kapaliny kontejneru.

2. **Poměr stran mřížky (kontejneru)** `[GRID_RATIO]`
   Poměr stran mřížky určuje tvar kontejneru obsahujícího kapalinu. Implicitní nastavení je poměr stran ($x = 1 : y = 1 : z = 1$). Aby byl zachován poměr objemu kapaliny a objemu kontejneru, který je při implicitním poměru stran, je velikost stran mřížky přepočítána následovně. Předpokládejme, že poměr stran je zadán čísly x, y a z a zadaná velikost mřížky je s_g. Pak se postupuje takto: $scale = \sqrt[3]{\frac{1}{(x \cdot y \cdot z)}}$. Výsledné velikosti stran, které by při implicitním nastavení byly (s_g, s_g, s_g), se vypočtou takto: $(s_x, s_y, s_z) = scale \cdot s_g \cdot (x, y, z)$.

3. **Rychlostní limit** `[VELOCITY_LIMIT]`
   Rychlostní limit jsem do simulace zavedl jako pokus o umělé tlumení některých výraznějších nestabilit. Přestože v ideálním případě nestability nevznikají vůbec, při použití konstantní velikosti simulačního kroku, který je kvůli interaktivitě co největší, se občasným nestabilitám v podobě vystřelených částic nelze vyhnout.

4. **Tuhost překážek** `[BOUNDARY_STIFFNESS]`
   Tvrdost překážky určuje velikost síly, kterou jsou částice odpuzovány.

5. **Tlumení na překážkách** `[BOUNDARY_DAMPENING]`
   Tlumení na překách kompenzuje vznik příliš velkých odpudivých sil.

6. **Měřítko simulace** `[SIMULATION_SCALE]`
   Měřítko simulace určuje společně s velikostí mřížky poměr objemu kapaliny a kontejneru, ve kterém je kapalina obsažena.

Nastavení simulace a vizualizace

Tyto parametry ovládají průběh simulace. Některé z nich lze ovládat interaktivně.

1. **Časový krok** `[TIMESTEP]`
   Jeden z nejdůležitějších parametrů vůbec. Volba správné velikosti simulačního kroku je klíčová pro interaktivní a přesto stabilní simulaci. Faktory ovlivňující časový krok jsou zmíněny na jiných místech této práce. Parametr lze měnit během simulace.

2. **Simulační scéna** `[SCENARIO]`
   Na výběr je několik ukázkových simulačních scén. Během simulace lze interaktivně měnit simulační scénu a simulaci resetovat.

3. **Barevné schéma** `[COLORING]`
   Určuje barevný gradient zobrazovaných částic. Na výběr jsou tato schémata: White, Blackish, BlackToCyan, BlueToWhite, HSVBlueToRed. Lze interaktivně měnit.

4. **Zdroj obarvení** `[COLOR_SOURCE]`
   Určuje zdroj gradientu obarvení. Možnosti jsou: rychlost, tlak a síla působící na částice (Velocity, Pressure, Force). Lze interaktivně měnit.

5. **Zařízení** `[DEVICE]`
   Číslo grafické karty (nebo jiného CUDA-enabled zařízení), na které bude simulace spuštěna. Pokud číslo neodpovídá, je zařízení vybráno automaticky.

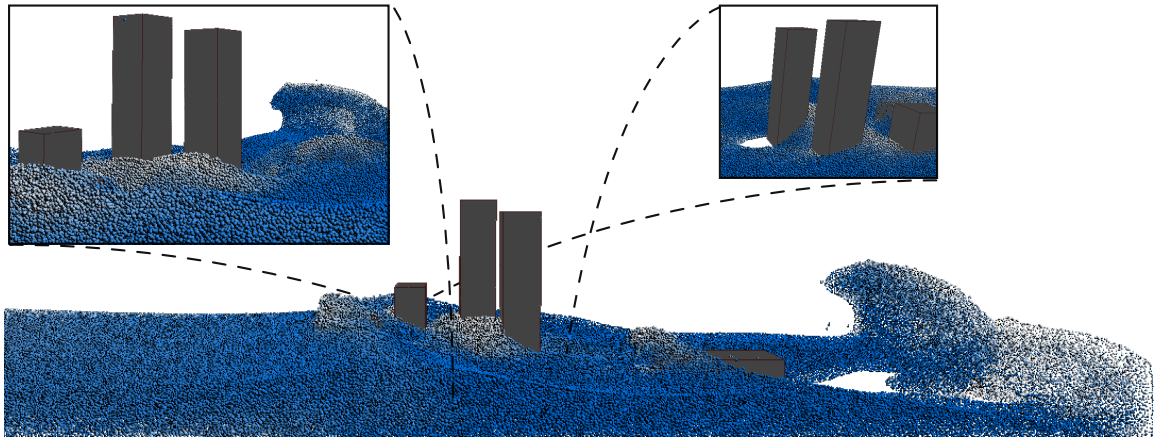
6. **Měření rychlosti** `[BENCHMARK]`
   Přepíná mezi módem měření rychlosti a vizualizací.

7. **Počet kroků simulace** `[BENCHMARK_ITER]`
   Počet kroků při měření rychlosti.

8. **Testovací výpisy** `[BENCHMARK_PRINT]`
   Určuje typ výpisů při testování.

Kapitola 6

Výsledky a zhodnocení



Obrázek 6.1: Obrázek z demonstrační aplikace

6.1 Volba parametrů simulace

Pro testování simulační aplikace jsem zvolil parametry, které jsou uvedeny v tabulce 6.1. Simulovaná kapalina s tímto nastavením odpovídá svým chováním přibližně vodě, tak jak ji pozorujeme v reálném životě. Zvolil jsem dostatečně malý časový krok, aby byly stabilní i simulace s velkým počtem částic.

Jak je popsáno v části 5.3.6, plynová konstanta určuje stlačitelnost simulované kapaliny. Pro testování jsem zvolil relativně malou hodnotu, protože větší hodnoty, které dávají vizuálně realističtější výsledky, mají kvůli stabilitě větší nároky na délku časového kroku.

6.1.1 Objem simulované kapaliny

Testování rychlosti aplikace jsem prováděl pro veliké rozmezí počtu částic (desítky tisíc až milióny). Zvýšení počtu částic ovšem znamená i zvětšení objemu simulované kapaliny. Tento jev není žádoucí, protože je vhodné, aby se kapalina s jakýmkoliv počtem částic chovala stejně. Počet částic o hmotnosti m v kapalině s klidovou hustotou ρ a daném objemu V lze

Parametr	Hodnota
časový krok	0.0005
klidová hustota	1000
klidový tlak	0
tvrdost překážky	25000
tlumení překážky	256
viskozita	1
plynová konstanta	2
měřítko simulace	0.001
velikost hrany mřížky	512

Tabulka 6.1: Parametry testování simulace

spočítat takto

$$n = \rho \frac{V}{m} \tag{6.1}$$

Z této rovnice lze snadno vyjádřit vztah pro objem simulované kapaliny tak, že je zřejmé, že objem kapaliny je přímo úměrný počtu částic, kterými je kapalina reprezentována. Tento vztah uvádím s dosazenými hodnotami, n = 131072:

$$V_{131072} = \frac{1}{\rho} nm = \frac{1}{1000} \frac{kg}{m^3} \cdot 0.00002kg \cdot 131072 = 0.00262144m^3 \tag{6.2}$$

Po dosazení $n = 32768$, tedy čtyřikrát menšího počtu částic, vychází objem kapaliny také čtyřikrát menší.

$$V_{32768} = \frac{1}{\rho} nm = \frac{1}{1000} \frac{kg}{m^3} \cdot 0.00002kg \cdot 32768 = 0.00065536m^3 \tag{6.3}$$

Tento problém řeším kompenzací hmotnosti částic. Při zdvojnásobení počtu částic zmenším hmotnost částic na polovinu. Tato úprava hmotnosti sice zachová objem simulované kapaliny, ale je také třeba upravit některé další parametry simulace, viz tabulka 6.2, které se automaticky přepočítávají.

Parameter/Počet částic	32k	128k	512k	2M
Hmotnost	0.0008	0.0002	0.00005	0.0000125
Klidová vzdálenost	0.00807636	0.00508779	0.00320511	0.00201909
Vzdálenost od přepážky	0.00403818	0.0025439	0.00160255	0.00100955
Vyhlazovací délka	0.0161527	0.0101756	0.00641021	0.00403818
Velikost hrany buňky	16.1527	10.1756	6.41021	4.03818

Tabulka 6.2: Změna parametrů v závislosti na změně hmotnosti částice a počtu částic. Počty částic jsou uváděny v mocninách 2. 32k tedy znamená $32 * 1024$ a 2M je $2 * 1024^2$

6.2 Měření rychlosti

Protože jedním z hlavních cílů této práce je pokusit se implementovat interaktivní simulátor kapalin, je rychlost aplikace velmi důležitým měřítkem úspěšnosti. Měření rychlosti

aplikace jsem rozdělil do několika kroků. Porovnával jsem časovou náročnost jednotlivých fází výpočtu 6.2.3, srovnával jsem rychlost aplikace na GPU, CPU a na CPU s více vlákny 6.2.4, rychlost na různých grafických kartách 6.2.2 a nakonec uvedu srovnání s jinými implementacemi 6.2.5.

Cílem měření je určit rychlost mé implementace algoritmu SPH pro simulaci viskózních kapalin. Proto jsem do měření rychlosti nezahrnul vizualizaci simulace, která není vzhledem k rychlosti algoritmu SPH podstatná. Výstupem měření je většinou počet iterací za sekundu. Měření času se spouští interně uvnitř programu, není do něj zahrnuta inicalizace modelu (sestavení testovací scény, alokace paměti atd.).

6.2.1 Testovací konfigurace

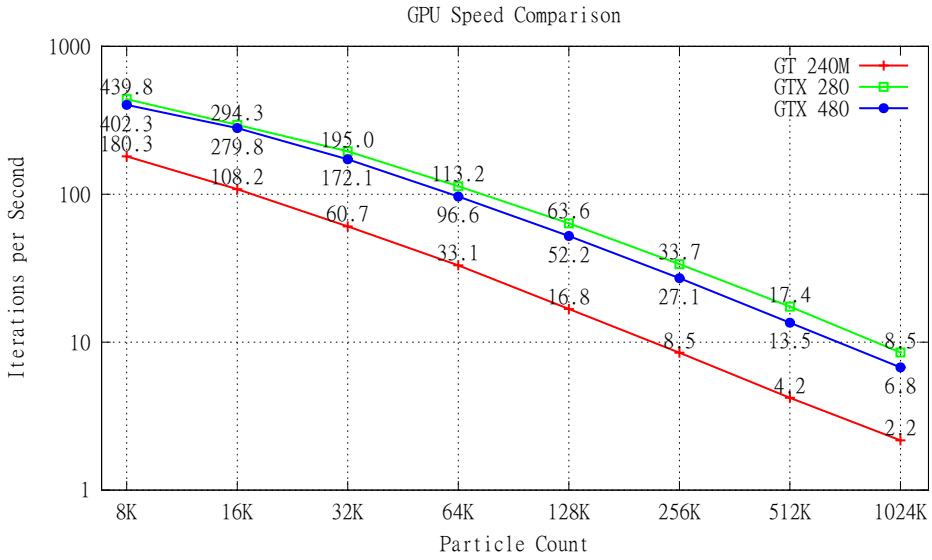
Testování jsem prováděl na notebooku Asus N61V s čtyřjádrovým procesorem Intel(R) Core(TM)2 Quad Q9000 s 4GB operační paměti DDR3 RAM pod 64bitovým operačním systémem Windows 7 a s grafickou kartou GeForce GT240M. Pokud není uvedeno jinak, všechny testy GPU implementace jsou prováděny na výše uvedené konfiguraci. K dalším testům jsem využíval ještě grafické karty GeForce GTX 280 a kartu GeForce GTX 480 s novou Fermi architekturou. Vlastnosti jednotlivých karet a verze použitého CUDA frameworku jsou uvedeny v tabulce 6.3

Atribut	GT 240M	GTX 280	GTX 480
CUDA Driver & Runtime :	3.20	3.20	3.20
CUDA Capability :	1.2	1.3	2.0
Global memory[bytes]:	1034485760	1017839616	1543176192
Multiprocessors:	6	30	15
Number of Cuda Cores:	48	240	480
Constant memory:	65536 bytes	65536 bytes	65536 bytes
Shared memory per block:	16384 bytes	16384 bytes	49152 bytes
Registers per block:	16384	16384	32768
Warp size:	32	32	32
Threads per block:	512	512	1024
Dimension of a block:	512 x 512 x 64	512 x 512 x 64	1024 x 1024 x 64
Dimension of a grid:	65535 x 65535 x 1	65535 x 65535 x 1	65535^3
Texture alignment:	256 bytes	256 bytes	512 bytes
Clock rate:	1.21 GHz	1.30 GHz	1.40 GHz
Estimated GFLOPS:	174	933	1344

Tabulka 6.3: Vlastnosti grafických karet GeForce použitých k testům

CUDA event API

Všude tam, kde to bylo z praktických důvodů vhodné, jsem k měření času použil CUDA event API, což je nejpřesnější způsob měření času na GPU, protože odpadá nutnost synchronizace s CPU.



Obrázek 6.2: Porovnání výkonu na různých grafických kartách

6.2.2 Srovnání výkonu na různých GPU

Rychlost implementace pro grafické karty není závislá pouze na počtu jader a frekvenci na které karta pracuje. Program je potřeba přeložit pro správnou generaci karet. Velké rozdíly se vyskytují u nových karet s technologií Fermi (GeForce GTX 480) s CUDA capability 2.0. U této nové řady splečnost NVIDIA zdvojnásobila počet tranzistorů na kartě a přibyly dvě nové L1 a L2 cache. Fermi architektura dále implementuje přesnější reprezentaci čísel typu float, takže některé programy určené pro starší karty se mohou chovat poněkud jinak.

Při porovnání rychlosti na třech testovacích kartách byla předpokládaným favoritem karta GeForce GTX 480, u které se počet operací v plovoucí desetinné čárce za vteřinu odhaduje na 1344. Oproti mému očekávání ovšem program pracuje rychleji na kartě GeForce GTX 280 (933 GFLOPS). Výsledky porovnání jsou na grafu6.2.

Z grafu lze vyčíst, že časová složitost algoritmu je lineární. Jak uvádí Müller [23], složitost SPH algoritmu lze zhruba vyjádřit jako $O(nm)$, kde n je počet částic a m je průměrný počet částic v okolních buňkách mřížky (jedná se o složitost samotného výpočtu SPH, nejsou započítány kroky hash, sort, update a integration, viz následující část).

Překvapivý nedostatek ve výkonu na kartě GTX 480 byl další motivací k bližšímu prozkoumání časové náročnosti jednotlivých fází algoritmu v následující části. Šlo zejména o identifikaci problematických částí algoritmu.

6.2.3 Srovnání jednotlivých fází výpočtu

Z důvodů optimalizace každé aplikace je vhodné zjistit časovou náročnost jednotlivých fází, kterými aplikace prochází. V našem případě se jedná se o tyto fáze výpočtu:

- hash — Výpočet hash hodnoty (příslušnost do buňky mřížky) pro každou částici
- sort — Setřídění indexů částic podle jejich hash hodnoty (algoritmus cudpp radix sort)

- update — V tomto kroku se podle setříděných indexů setřídí ostatní data částic (pozice, rychlosti…)
- density (SPH step 1) — Aplikací metody SPH se pro každou částici spočítá hustota a ze stavové rovnice se vyjádří příslušný tlak.
- forces (SPH step 2) — Díky znalosti hustoty a tlaku lze nyní vyčíslit všechny síly působící na částici.
- integration — Integrace sil, reakce na kolize s překážkami

Časy trvání jednotlivých fází pro všechny testované počty částic jsou shrnuty v tabulce 6.4 (GT 240M).

částic	hash	radix sort	update dat	výpočet hustoty	výpočet sil	integrace sil
1K	0.0215	0.1533	0.0281	0.3112	0.6441	0.0375
2K	0.0264	0.2167	0.0362	0.3649	0.6759	0.0480
4K	0.0309	0.2514	0.0516	0.7324	1.2380	0.0783
8K	0.0467	0.4505	0.0941	1.2574	2.1001	0.1252
16K	0.0675	0.6590	0.1649	2.3358	4.1823	0.2002
32K	0.1099	0.9222	0.2858	4.7789	8.1763	0.3477
64K	0.1942	1.6564	0.4820	9.3015	15.5905	0.5813
128K	0.3620	3.4902	1.1030	18.9231	31.2713	1.1252
256K	0.7022	6.2708	2.1535	38.6440	63.5326	2.1627
512K	1.3801	14.4200	4.2071	78.0613	129.4028	4.2975
1024K	2.7111	28.5200	10.5862	149.3061	249.8794	11.0293
%	1.00%	8.68%	2.12%	31.41%	54.22%	2.57%

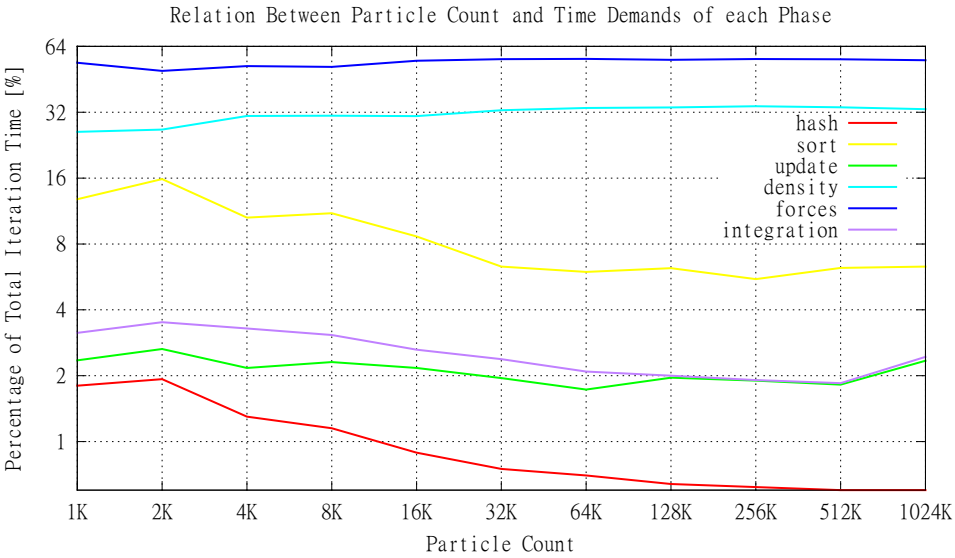
Tabulka 6.4: Časová náročnost jednotlivých fází simulace na GT 240M. V jednotlivých sloupcích je čas potřebný ke spočtení dané fáze v milisekundách

Jak je vidět v tabulce 6.4 a v grafu 6.3, poměrná časová náročnost jednotlivých částí algoritmu zůstává při měnícím se počtu částic přibližně stejná. Dále je vidět že zdaleka nejvíce času v jednom časovém kroku zabírá výpočet sil metodou SPH, těsně následován výpočtem hustoty. To jsou části výpočtu, na které je třeba se nejvíc zaměřit při případné optimalizaci.

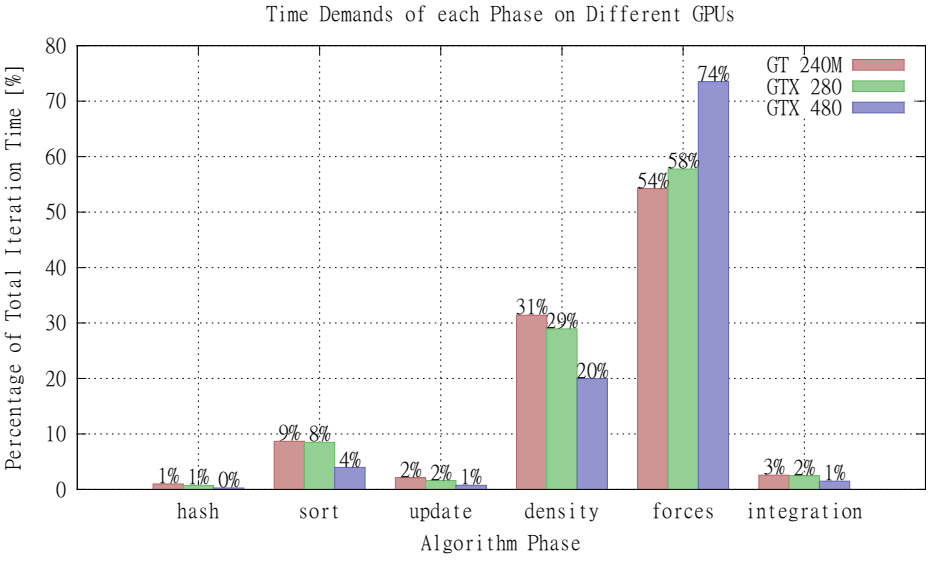
Různé GPU

Obecné charakteristiky algoritmu uvedené výše platí na všech použité grafické karty. Procentuální srovnání trvání jednotlivých fází výpočtu na různých kartách je přehledně zobrazeno v grafu 6.4. Z grafu lze snadno vyčíst, že při použití karty GeForce GTX 480 zabírá výpočet sil metodou SPH průměrně 74 procent času jedné iterace. Poměrně trvá na této kartě (oproti ostatním testovaným) tedy výpočet sil daleko víc. Pro lepší ilustraci je na grafu 6.5 porovnání absolutních časů.

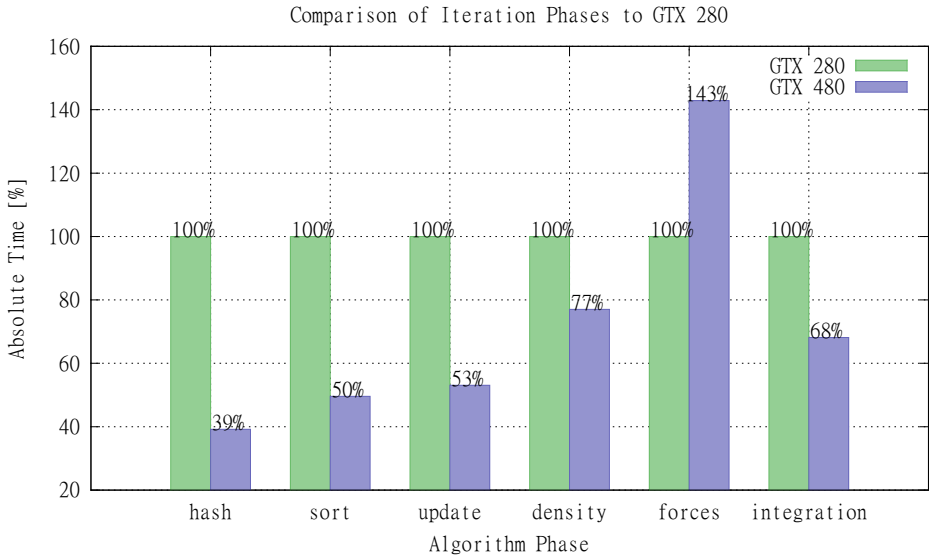
Na grafu 6.5 je vidět, že všechny části algoritmu s vyjímkou výpočtu sil běží na kartě GTX 480 rychleji než na kartě GTX 280. Je zajímavé, že to platí i pro fázi výpočtu hustot, která principielně pracuje stejně jako výpočet sil. Rozdílem mezi těmito fázemi je větší počet přístupů do paměti ve fázi výpočtu sil, který, zdá se, na architektuře fermi výpočet zpomaluje.



Obrázek 6.3: Srovnání jednotlivých částí výpočtu na GT 240M v závislosti na počtu částic



Obrázek 6.4: Srovnání trvání jednotlivých fází výpočtu na různých grafických kartách



Obrázek 6.5: Porovnání absolutního času jednotlivých fází s časy na kartě GTX 280.

6.2.4 Srovnání GPU a CPU implementace

V rámci měření rychlosti simulace na grafickém procesoru jsem se rozhodnul provést srovnání rychlostí implementace na GPU a CPU. Testování jsem prováděl s konfigurací uvedenou na začátku této sekce.

Převod implementace GPU na CPU se ukázal jako poměrně snadný. Protože k dispozici mám počítač s čtyřjádrovým procesorem, rozhodl jsem se navíc ještě implementovat verzi simulátoru využívající všechna 4 jádra. K implementaci pro více vláken jsem použil OpenMP. Výsledky porovnání rychlosti jsou uvedeny v tabulce 6.5 a v grafu 6.6.

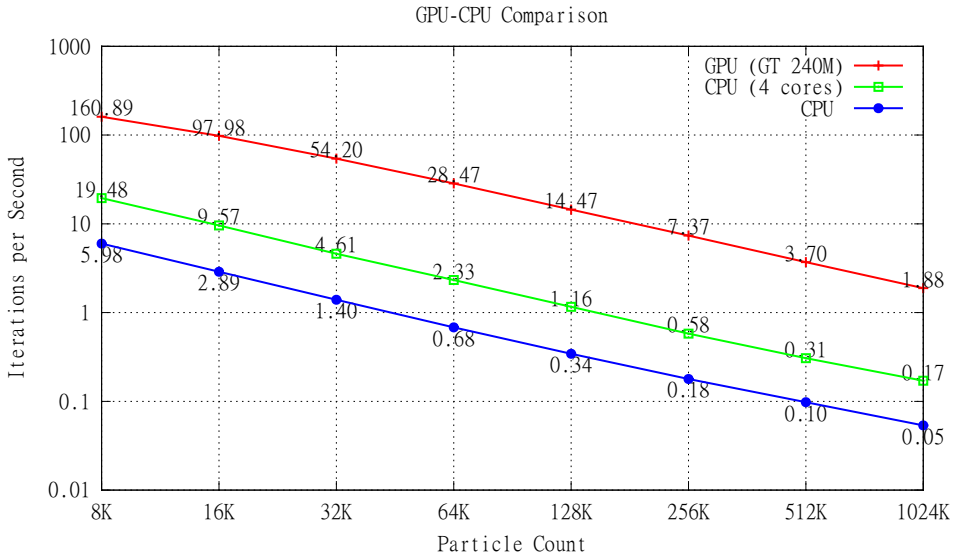
Implementace	8K	16K	32K	64K	128K	256K	512K	1024K
GPU (GT 240M)	160.88	97.97	54.20	28.47	14.47	7.36	3.69	1.88
CPU 1 jádro	19.47	9.57	4.61	2.32	1.16	0.57	0.30	0.17
CPU 4 jádra	5.98	2.88	1.39	0.68	0.34	0.17	0.09	0.05

Tabulka 6.5: Rychlosti jednotlivych implementací, hodnoty udávají počet iterací za sekundu

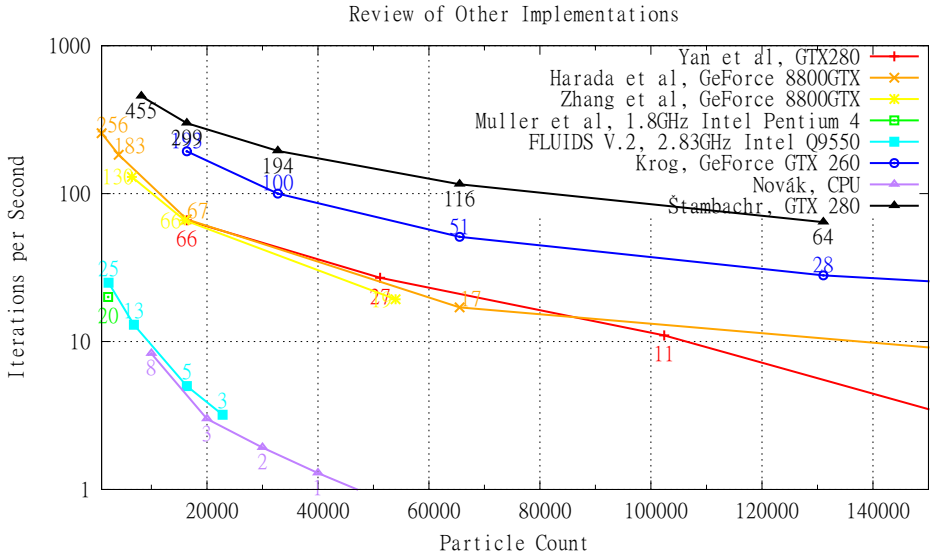
Implementace pro CPU nelze podle očekávání použít k interaktivní simulaci systémů skládajících se z více částic. Oproti tomu implementace pro 4 jádra je na tom o něco lépe, vykazuje zrychlení (cca 3.5krát) a šlo by interaktivně simulovat systémy s počtem částic okolo osmi tisíc.

6.2.5 Srovnání s jinými implementacemi

Vytvořit přesné srovnání výkonu s jinými implementacemi je poměrně náročný úkol, protože většina autorů neuvádí přesné parametry, které použili pro testování svých modelů. Dalším důvodem je jistá odlišnost jednotlivých modelů a v neposlední řadě také odlišnost hardware, na kterém jednotliví autoři testy prováděli. Přesto se pokusím takové srovnání vytvořit, aby čtenář získal hrubý přehled o výkonu dosaženém jinými autory. Je třeba zmínit, že tento přehled zahrnuje jak implementace pro CPU, tak implementace pro GPU.



Obrázek 6.6: Srovnání rychlosti implementace pro GPU, jednovláknové implementace a vícevláknové implemntace běžící na 4 jádrech



Obrázek 6.7: Srovnání implementací jiných autorů

GPU implementace

Má implementace si nevede ve srovnání s dřívějšími implementacemi špatně. Přímé srovnání se nabízí s Yan et al [37], protože jsem měl na testování k dispozici stejnou grafickou kartu (GeForce GTX 280). Z grafu 6.7 lze vyčíst, že má implementace je až 5krát rychlejší. Abych byl v hodnocení spravedlivý, je třeba uvést, že Yan et al. používají výpočetně složitější verzi algoritmu a pravděpodobně měřil čas i s renderováním.

Harada et al. [12] testoval svou implementaci na starší architektuře GPU (GeForce 8800GTX), s výsledkem 17 iterací za vteřinu pro 64K částic. Na stejné kartě testovali svou implementaci algoritmu SPH i Zhang et al. [38] a dosáhli podobných výsledků. Je třeba poznamenat, že nejlepší rychlosti, se kterou jsem se setkal, dosáhl Krog [14] s kartou GTX 470 (není znázorněno v grafu).

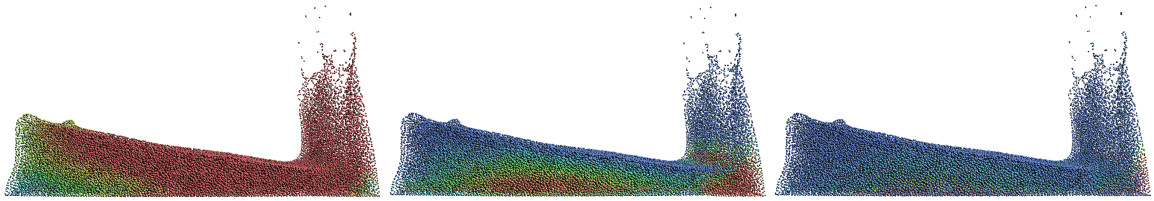
CPU implementace

Zde můžu uvést srovnání s mou vlastní CPU implementací. Müller [23] implementoval svůj SPH model na počítači s 1.8GHz Intel Pentium 4 procesorem, dosáhl realtime výsledku 20ips pro 2200 částic. Krog [14] testoval optimalizovaný framework FLUIDS V.2 na počítači s procesorem 2.83GHz Intel Q9550 a dosáhl o něco lepších výsledků, viz graf 6.7. Novák [25] dosáhl ve své práci výsledků horších, jeho model ovšem nebyl určen pro interaktivní aplikace.

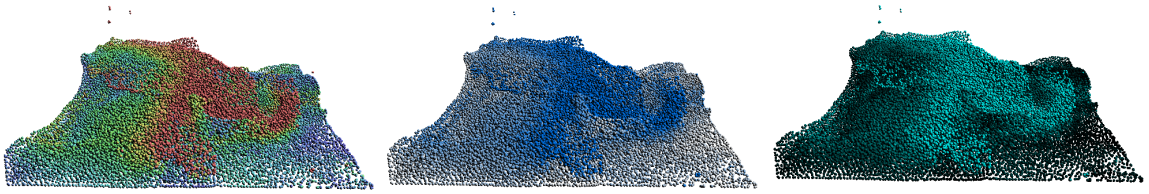
6.3 Vizuální výstup

Mým cílem bylo vytvořit několik testovacích scén, na kterých lze demonstrovat použitelnost modelu pro simulování kapalin. Jak jsem již psal v kapitole 3.4, jednou z největších nevýhod metody SPH je obtížná extrakce povrchu, které je potřeba pro realistickou vizualizaci. Tato extrakce a následná vizualizace kapaliny lze provést např. metodou Marching Cubes a pomocí techniky point splatting. Vizualizace povrchu těmito technikami je ovšem náročná, např. Novák [25] uvádí, že algoritmus Marching Cubes má kvadratickou složitost. Proto je v současné době nemožné provádět takovéto renderování v reálném čase.

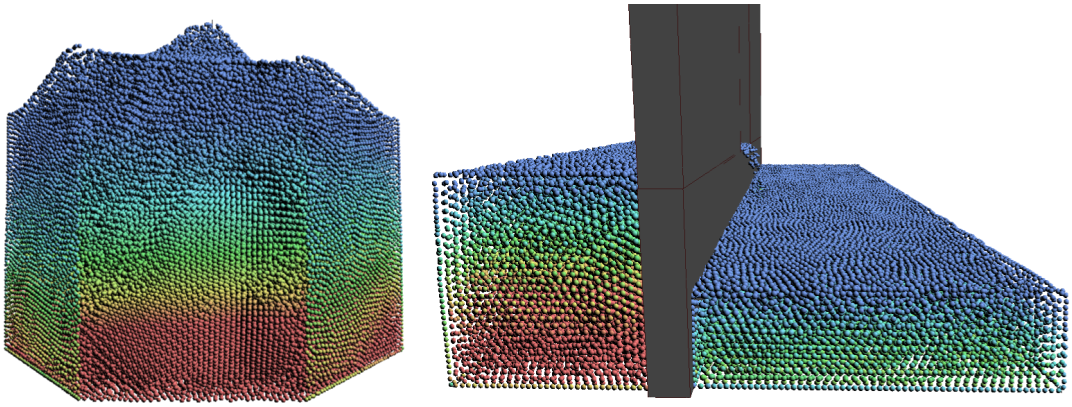
V této práci jsem implementoval po vzoru Greena [11] jednoduchou vizualizaci. Každá částice v systému je zobrazena jako koule vystínovaná pomocí vertex a pixel shaderu. Barva každé částice reflektuje určitou veličinu příslušnou dané částici. Tyto veličiny jsou: rychlost, síla působící na částici a tlak a jsou zobrazeny na obrázku 6.8. V programu jsou implementovány 3 druhy zabarvení částic, viz obrázek 6.9.



Obrázek 6.8: Zobrazení různých stavových veličin, zleva: rychlost, tlak, síla



Obrázek 6.9: Rychlost částic zobrazená třemi barevnými schématy, která jsou v aplikaci k dispozici



Obrázek 6.10: Znázornění tlaku v kapalině. Vlevo: Řez objemem kapaliny odhaluje dobře patrný deficit tlaku na hranicích kapaliny a kontejneru vyplývající z podstaty metody SPH a použitých hraničních podmínek. Vpravo: Rozdíl tlaku v závislosti na výšce vodního sloupce.

6.3.1 Testovací scény

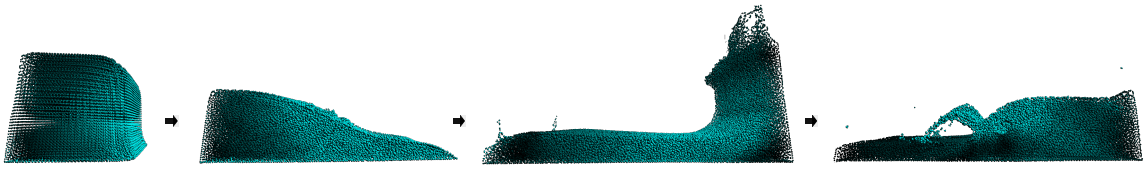
V průběhu vývoje aplikace jsem vytvořil několik testovacích scén. V této sekci jsou některé popsány a jsou u nich uvedeny ilustrující obrázky.

Scéna 1

Jedná se o základní simulaci prolomení přehrady (Dam break). Kapalina je inicializována do tvaru kvádru na jedné straně kontejneru. Scéna je zobrazena na obrázku 6.11.

Scéna 2

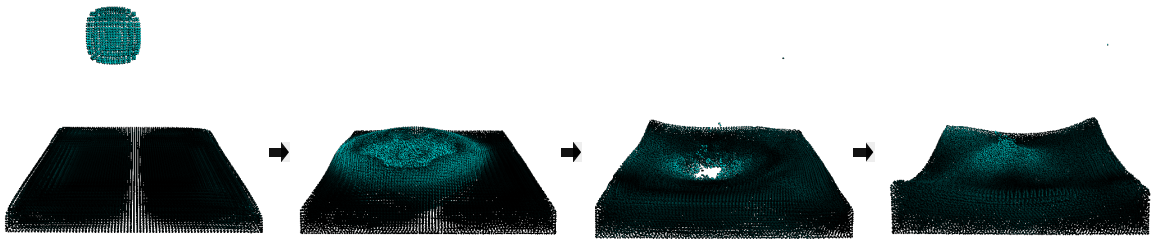
Jedná se o scénu, která je dost podobná první scéně. Rozdíl je, že kapalina je umístěná do rohu kontejneru a po jejím uvolnění se rozlévá do dvou stran.



Obrázek 6.11: Testovací scéna 1

Scéna 3

Malé množství kapaliny je puštěno volným pádem do nádoby, ve které je plytká vrstva kapaliny.



Obrázek 6.12: Testovací scéna 3

Scéna 4

Celý objem kapaliny je puštěn volným pádem do kontejneru.

Scéna 5

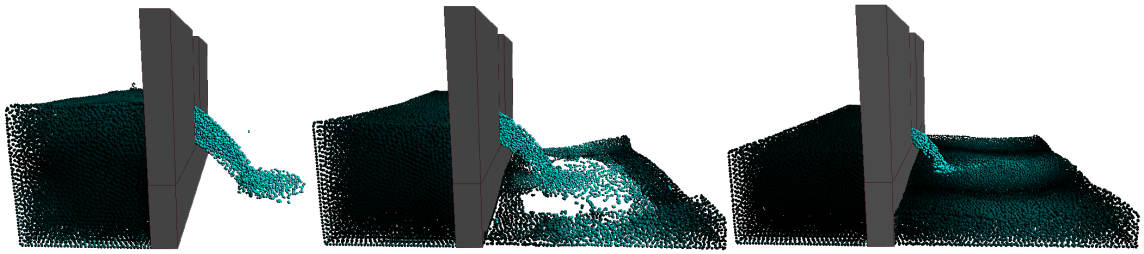
Kapalina je vygenerována po celém obsahu podstavy kontejneru a je pokud možno v rovnovážném stavu. Pokud je ovšem podstava kontejneru malá a tudíž velká hloubka kapaliny, kapalina se musí ustálit.

Scéna 6

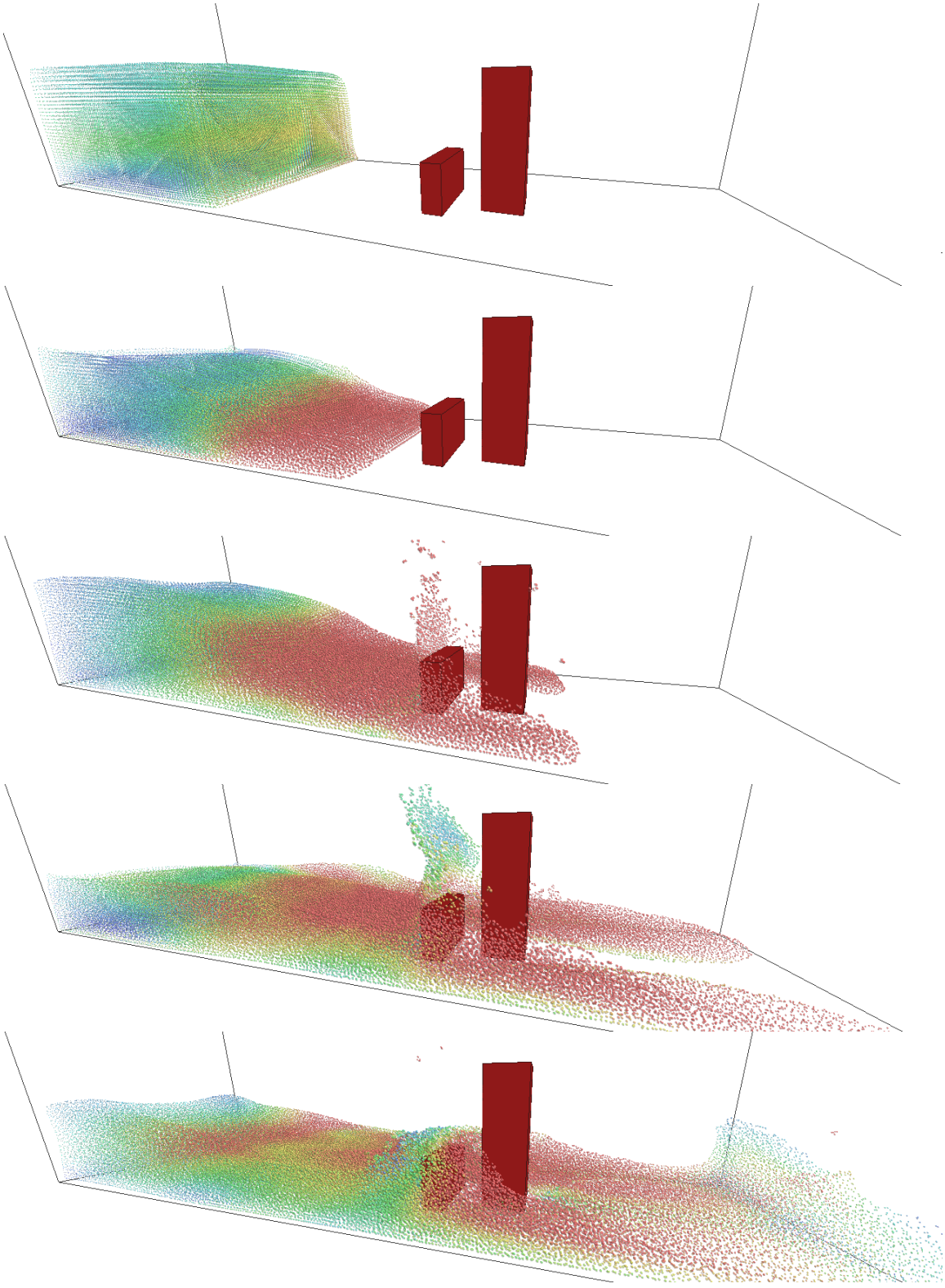
Simulace zatopení několika domů vodou z prolomené přehrady.

Scéna 7

Kapalina je umístěna v přehradě, ze které voda otvorem vytéká. Na této scéně je dobře patrný vliv konstanty tuhosti. Při velké tuhosti kapaliny je třeba volit malý časový krok, v opačném případě hrozí, že kapalina začne z přehrady unikat „skrz“ zdi. Scéna je na obrázku 6.13.



Obrázek 6.13: Testovací scéna 7 - přehrada



Obrázek 6.14: Tlumení vlny pomocí překážky

Kapitola 7

Závěr

V průběhu této práce jsem popsal proces vedoucí k sestrojení interaktivního simulátoru kapalin. Simulátor pracuje na principu Smoothed Particle Hydrodynamics, částicové interpolační lagrangeovské metody. Metodu SPH jsem v kombinaci s lagrangeovskou formulací Navier-Stokesových rovnic aplikoval na problém simulace newtonských kapalin s nízkou viskozitou (kapaliny blízké vodě). V rámci implementace jsem se zaměřil na rychlost simulace na úkor fyzikální přesnosti. Proto jsem kladl důraz na použití nejméně výpočetně náročných vyhlazovacích funkcí a algoritmů pro vyhledávání blízkých částic.

Grafické karty v dnešní době přestávají sloužit pouze k zobrazování, ale stále více jsou využívány k obecným a fyzikálním výpočtům. Nové generace grafických karet dokonce umožňují využívat čísel typu double. Vzhledem k částicové povaze metody SPH, je její paralelizace poměrně snadná. Proto bylo nasnadě ji implementovat pro grafické karty firmy NVIDIA s využitím technologie CUDA. V rámci testování rychlosti aplikace jsem vytvořil dvě verze programu pro CPU (pro jedno i více jader). Srovnáním výkonu těchto implementací je zřejmý obrovský rozdíl ve výpočetním výkonu ve prospěch paralelních výpočtů na grafických kartách. Porovnání s impementacemi pro grafické karty jiných autorů je problematické (zejména z důvodu rozlišnosti použitého hardware), přesto mohu říci, že rychlost mého programu v tomto srovnání nezaostává.

Validace modelu probíhala pouze na základě posouzení vizuálního výstupu a jeho porovnání s reálným chováním kapalin — tedy na základě vizuální věrohodnosti. Přesto si dovolím tvrdit, že při správném nastavení parametrů je implementovaný algoritmus vhodný k interaktivní simulaci kapalin podobných vodě. Neduhem metody SPH může lze považovat fakt, že se metoda je formulována pro stlačitelné tekutiny a je snaha ji použít pro téměř nestlačitelné kapaliny. Nestlačitelnost se tedy uvozuje do modelu uměle, což má za následek vzájemně se vylučující jevy: rychlost simulace a nestlačitelnost kapaliny. Čím více je kapalina stlačitelná, tím větší časový krok je možno použít a naopak. Dalším nedostatkem je potřeba velkého počtu částic pro získání detailního povrchu.

7.1 Další vývoj

Simulační aplikaci lze s poměrně malým úsilím upravit, aby využívala alternativní přístupy k simulaci kapalin metodou SPH. Lze snadno přidat jiné vyhlazovací funkce, které umožní přesnější (ale pomalejší) výpočty a zavést z literatury známé korekce pro laplacián a gradient vyhlazovacích jader. Dále je možné implementovat lepší způsob reakcí na kolize např. za využití stínových částic, které přímo přispívají do řešení SPH. Poměrně snadné by bylo

přidání terénů a výškových map. Je určitě možné aplikaci modifikovat pro simulaci nenewtonských kapalin, to by ovšem vyžadovalo použití složitějšího reologického modelu.

Zřejmý je prostor pro vylepšení v oblasti vizualizace. Zde bohužel metoda SPH, kvůli obtížné extrakci povrchu, zaostává. Vizuálně přitažlivější vizualizace povrchu lze dosáhnout v režimu offline např. za použití algoritmů Marching Cubes a Point Splatting. Jak takovou vizualizaci provést v reálném čase mi není známo.

Aplikaci jsem implementoval pro grafické karty firmy NVIDIA, s podporou technologie CUDA. Do budoucna by ovšem bylo lepší implementaci převést na standard OpenCL, který umožňuje použití širšího rozmezí grafických karet.

Algoritmus je velmi náchylný na správné nastavení parametrů a určitě by benefitoval z další práce na nastavení počátečních podmínek, např. inicializace kapaliny do klidového stavu.

Literatura

[1] Amada, T.: Real-time particle based fluid simulation with rigid body interaction. *In Game Programming Gems*, ročník 6, 2006: s. 189–205.

[2] Attwood, R. E.; Goodwin, S. P.; Whitworth, A. P.: Adaptive smoothing lengths in SPH. *Astronomy and Astrophysics*, ročník 464, Březen 2007: s. 447–450.

[3] Bridson, R.; Müller-Fischer, M.: Fluid simulation: SIGGRAPH 2007 course notes. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, 2007, s. 1–81.
URL `<http://www.cs.ubc.ca/~rbridson/fluidsimulation/fluids\_notes.pdf>`

[4] Chen, S.; Doolen, G. D.; Eggert, K. G.: Lattice-Boltzmann Fluid Dynamics. *Los Alamos Science*, ročník 22, č. 1, 1994: s. 98 – 109.

[5] Chorin, A.; Marsden, J.: *A mathematical introduction to fluid mechanics*. Texts in applied mathematics, Springer-Verlag, 1993, ISBN 9780387979182.

[6] Cleary, P. W.; Prakash, M.: Discrete-element modelling and smoothed particle hydrodynamics: potential in the environmental sciences. *hilosophical Transactions of the Royal Society - Series A: Mathematical, Physical and Engineering Sciences*, ročník 362, č. 1822, 2004: s. 2003–2030.
URL `<http://www.ncbi.nlm.nih.gov/pubmed/15306427>`

[7] Crespo, A. J. C.: *Application of the Smoothed Particle Hydrodynamics model SPHysics to free-surface hydrodynamics*. Dizertační práce, University of Vigo, 2008.

[8] Dalrymple, R. A.: *Particle Methods and Waves, with Emphasis on SPH*. Johns Hopkins University, Červen 2007.

[9] Desjardins, B.; Chi-Kun, L.: A Survey of the Compressible Navier–Stokes Equations. *Taiwanese Journal of Mathematics*, ročník 3, č. 2, Červen 1999: s. 123–137.

[10] Gingold, R. A.; Monaghan, J. J.: Smoothed particle hydrodynamics - Theory and application to non-spherical stars. *Royal Astronomical Society, Monthly Notices*, ročník 181, Listopad 1977: s. 375–389.

[11] Green, S.: Particle Simulation using CUDA. Technická zpráva, NVIDIA, 2010.

[12] Harada, T.; Koshizuka, S.; Kawaguchi, Y.: Smoothed particle hydrodynamics on GPUs. In *Computer Graphics International*, 2007, s. 63–70.

[13] Harris, M. J.: GPU Gems - Chapter 38. Fast Fluid Dynamics Simulation on the GPU. 2008-05-13 [cit. 2008-12-19].
URL `<http://http.developer.nvidia.com/GPUGems/gpugems\_ch38.html>`

[14] Krog, O. E.: *GPU-based Real-Time Snow Avalanche Simulations*. Diplomová práce, Norwegian University of Science and Technology, 2010.

[15] Li, S.; Liu, W. K.: *Meshfree Particle Methods*. Springer, první vydání, Srpen 2004, ISBN 3540222561.

[16] Liu, M. B.; Liu, G. R.; Lam, K. Y.: Constructing smoothing functions in smoothed particle hydrodynamics with applications. *J. Comput. Appl. Math.*, ročník 155, June 2003: s. 263–284, ISSN 0377-0427.
URL <http://dx.doi.org/10.1016/S0377-0427(02)00869-5>

[17] Lomax, H.; Pulliam, T. H.; Zingg, D. W.: *Fundamentals of Computational Fluid Dynamics (Scientific Computation)*. Springer, Červen 2001, ISBN 3540416072.

[18] Losasso, F.; Gibou, F.; Fedkiw, R.: Simulating water and smoke with an octree data structure. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, New York, NY, USA: ACM, 2004, s. 457–462.
URL <http://doi.acm.org/10.1145/1186562.1015745>

[19] Lucy, L. B.: A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*, ročník 82, Prosinec 1977: s. 1013–1024.

[20] Monaghan, J. J.: Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics*, ročník 30, 1992: s. 543–574.

[21] Monaghan, J. J.; Lattanzio, J. C.: A refined particle method for astrophysical problems. *Astronomy and Astrophysics*, ročník 149, Srpen 1985: s. 135–143.

[22] Morris, J. P.; Fox, P. J.; Zhu, Y.: Modeling Low Reynolds Number Incompressible Flows Using SPH. *Journal of Computational Physics*, ročník 136, č. 1, 1997: s. 214 – 226, ISSN 0021-9991.
URL <http://www.sciencedirect.com/science/article/B6WHY-45V7FSX-G/2/43b922e2a2ca81607847cf038c3afe49>

[23] Müller, M.; Charypar, D.; Gross, M.: Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, Červenec 2003, ISBN 1-58113-659-5, s. 154–159.
URL <http://portal.acm.org/citation.cfm?id=846298>

[24] Nelson, R. P.; Papaloizou, J. C. B.: Variable Smoothing Lengths and Energy Conservation in Smoothed Particle Hydrodynamics. *Royal Astronomical Society, Monthly Notices*, ročník 270, Září 1994: s. 1–20.
URL <http://arxiv.org/pdf/astro-ph/9406053v1>

[25] Novák, O.: *Simulace viskózních kapalin*. Diplomová práce, ČVUT, 2007.

[26] Peralta-Fabi, R.: On the Foundations of the Navier-Stokes Equations. *Revista Mexicana de Física*, ročník 37, č. 1, Červen 1991: s. 117–131.

[27] Shao, S.; Lo, Y. M. E.: Incompressible SPH method for simulating Newtonian and non-Newtonian flows with a free surface. *Advances in Water Resources*, ročník 26, Červenec 2003: s. 787–800.

[28] Stam, J.: Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999, ISBN 0-201-48560-5, s. 121–128.
URL <http://dx.doi.org/10.1145/311535.311548>

[29] Succi, S.: *The lattice Boltzmann equation for fluid dynamics and beyond*. Numerical mathematics and scientific computation, Clarendon Press ; Oxford University Press, Srpen 2001, ISBN 0198503989.

[30] Succi, S. and Sbragaglia, M and Ubertini S.: Lattice Boltzmann Method. 2010-12-02 [cit. 2008-12-21].
URL <http://www.scholarpedia.org/article/Lattice_Boltzmann_Methods>

[31] Takeda, H.; Miyama, S. M.; Sekiya, M.: Numerical Simulation of Viscous Flow by Smoothed Particle Hydrodynamics. *Progress of Theoretical Physics*, ročník 92, Listopad 1994: s. 939–960.

[32] Vlček, A.: *Real-time vizualizace povětrnostních vlivů v terénu*. Diplomová práce, VUT v Brně, 2009.

[33] Šťava, O.; Beneš, B.; Brisbin, M.; aj.: Interactive terrain modeling using hydraulic erosion. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '08, Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2008, ISBN 978-3-905674-10-1, s. 201–210.
URL <http://portal.acm.org/citation.cfm?id=1632592.1632622>

[34] Wagner, A. J.: *A Practical Introduction to the Lattice Boltzmann Method*. North Dakota State University, Březen 2008.
URL <http://www.ndsu.edu/fileadmin/physics.ndsu.edu/Wagner/LBbook.pdf>

[35] Wendland, H.: Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, ročník 4, č. 1, Prosinec 1995: s. 389–396.
URL <http://dx.doi.org/10.1007/BF02123482>

[36] WWW stránky: Fluid Dynamics and the Navier-Stokes Equations. 2010-12-02 [cit. 2008-12-19].
URL <http://universe-review.ca/R13-10-NSeqs.htm>

[37] Yan, H.; Wang, Z.; He, J.; aj.: Real-time fluid simulation with adaptive SPH. *Computer Animation and Virtual Worlds*, ročník 20, č. 2-3, 2009: s. 417–426, ISSN 1546-427X, doi:10.1002/cav.300.
URL <http://dx.doi.org/10.1002/cav.300>

[38] Zhang, Y.; Solenthaler, B.; Pajarola, R.: GPU accelerated SPH particle simulation and rendering. In *ACM SIGGRAPH 2007 posters*, SIGGRAPH '07, New York, NY, USA: ACM, 2007.
URL <http://doi.acm.org/10.1145/1280720.1280731>

Příloha A

Ovládání aplikace

Aplikace je pro operační systém Windows a vyřaduje nainstalovaný CUDA toolkit v. 3.2. Lze ji spouštět z příkazové řádky, což slouží zejména pro testovací účely, z výhodou toho lze použít v testovacích skriptech. Program při spuštění z příkazové řádky přijímá následující parametry:

- -pnum [počet částic]
- -iter [počet iterací pro měření rychlosti]
- -bprint [typ výpisů při měření rychlosti]
- -dev [číslo graf. karty s CUDA]

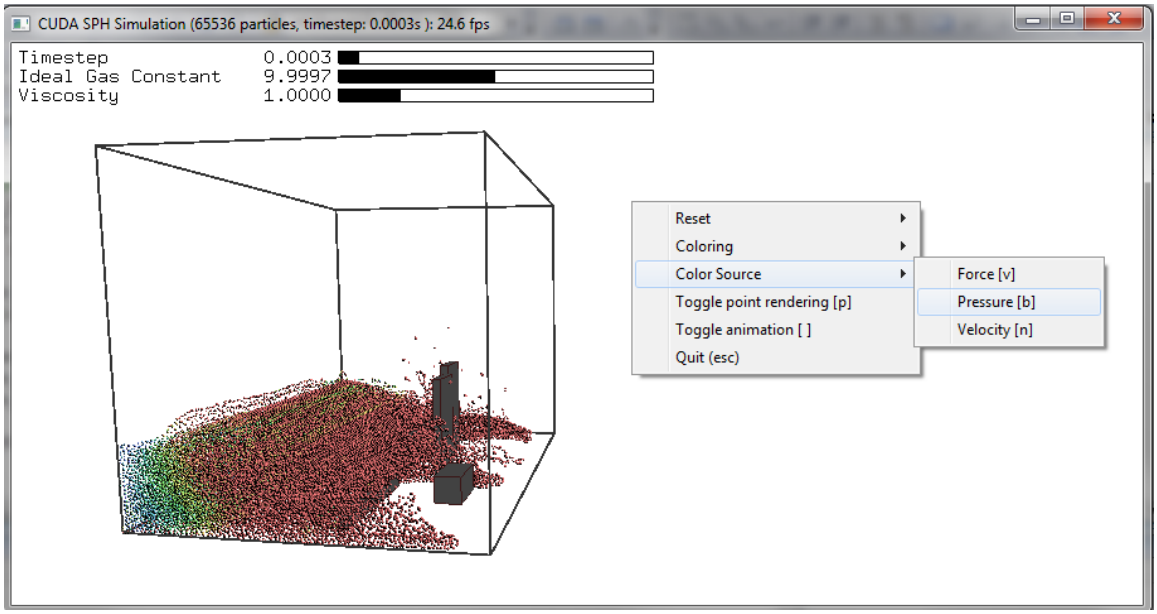
Všechny ostatní parametry lze zadat přes konfigurační soubor. Zde je třeba podotknout, že parametry zadané přes příkazovou řádku mají přednost před těmi načtenými z konfiguračního souboru. Ukázkový konfigurační soubor je v příloze B. Vysvětlení parametrů a jejich fyzikální význam je uveden v kapitole o implementaci 5.3.6.

Interaktivní ovládání aplikace

Interaktivní aplikace je realizována pomocí knihovny GLUT, která poskytuje pouze omezené možnosti tvorby GUI. Hlavní okno aplikace zobrazuje simulovanou scénu. Scénou lze manipulovat, měnit pohled takto:

- Stiskem levého a prostředního tlačítka a pohybem myši se scéna posouvá zepředu dozadu.
- Stiskem levého tlačítka při zmáčknuté klávese `Shift` a pohybem myši se scéna posouvá zleva doprava a nahoru dolů.
- Stiskem levého tlačítka a pohybem myši se otáčí scéna.
- Stiskem pravého tlačítka se vyvolá interaktivní menu.

Pomocí menu lze měnit použité barevné schéma, zdroj obarvení lze resetovat simulaci pro zvolenou scénu, zobrazit posuvníky ovlivňující další parametry simulace a zastavit simulaci. U všech položek menu je uvedena příslušející klávesová zkratka. Po zobrazení posuvníků je možné měnit nejdůležitější parametry simulace, těmi je:



Obrázek A.1: Ovládání interaktivní implementace

- Délka simulačního kroku - je možné vyzkoušet, kdy simulace přestane být stabilní.
- Tuhost kapaliny - možno vyzkoušet, jak „tuhá“ kapalina může být při současném nastavení simulačního kroku.
- Viskozita - vyšší hodnoty opět negativně ovlivňují stabilitu simulace.

Příloha B

Konfigurační soubor

Konfigurační soubor musí být umístěn v aktuálním adresáři, ze kterého je program spouštěn. Pokud program nenalezne konfigurační soubor, spustí se program s implicitním nastavením viz ukázkový konfigurační soubor.

Ukázkový konfigurační soubor

```
# fluid params
PARTICLES_NUMBER: 32768
GRID_WORLD_SIZE: 512
GRID_RATIO: 1:1:1       #x:y:z
TIMESTEP: 0.0005        #with current setting up to 0.002
REST_DENSITY: 1000      #kg.m-3, higher values requires lower timestep
IDEAL_GAS_CONSTANT: 2.0 #higher values requires lower timestep
VISCOSITY: 1.0          #mPa.s
BOUNDARY_STIFFNESS: 25000
BOUNDARY_DAMPENING: 256
VELOCITY_LIMIT: 500
SIMULATION_SCALE: 0.001
STATIC_FRICTION_LIMIT: 0
KINETIC_FRICTION: 0

# non-fluid params
SCENARIO: 2             # 1 through 7 for now
COLORING: HSVBlueToRed  # White, Blackish, BlackToCyan, BlueToWhite
COLOR_SOURCE: Velocity  # Pressure, Force

#simulation setup
DEVICE: 0               # device number
BENCHMARK: 0            # false = 0, true = 1
BENCHMARK_ITER: 1000    # benchmark iterations
BENCHMARK_PRINT: 0      # 0=speed, 1=parts, 2=memory, 3==grid, 4=all
```