České vysoké učení technické v Praze
Fakulta elektrotechnická

Diplomová práce

Simulace viskózních kapalin

Ondřej Novák

Vedoucí práce: Ing. Jaroslav Sloup

Studijní program: Výpočetní technika magisterský

Obor: Informatika a výpočetní technika

květen 2007

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 20.5. 2007 .....

Abstract

This thesis deals with simulation and visualization of viscous liquids. It describes theory, including base Navier-Stokes equations, but focus of this thesis lays in describing method of numerical solution of those equations by using SPH (Smoothed particle hydrodynamics) and in implementation of this method. By using SPH it is possible to simulate detailed small scale scenes at interactive framerates with usage of few thousands particles. Used visualization techniques allow not only displaying surface of liquid, but also visualise state variables of liquid model, which can be used to clarify model function.

Abstrakt

Tato práce se zabývá simulací a vizualizací viskózních kapalin. Je zde popsána teorie včetně výchozích Navier-Stokesových rovnic, těžiště práce je však v popisu jejich numerického řešení pomocí metody SPH (Smoothed particle hydrodynamics) a v implementaci této metody. Pomocí této metody je dosaženo detailní simulace kapaliny v malé scéně při interaktivní rychlosti za použití několika málo tisíc částic. Použité vizualizační postupy umožňují nejenom zobrazení povrchu kapaliny, ale také vizualizaci stavových veličin modelu kapaliny, což lze využít pro objasnění práce modelu.

Obsah

Seznam obrázků **ix**

Seznam tabulek **xi**

1 Úvod **1**
1.1 Cíl práce . . . 1
1.2 Členění práce . . . 1

2 Stav oboru **3**

3 Klasická výpočetní hydrodynamika **6**
3.1 Navier-Stokesovy rovnice . . . 6
3.2 Eulerova Metoda . . . 6

4 Smoothed particle hydrodynamics **8**
4.1 Definice . . . 8
4.2 Jádra . . . 9
4.2.1 Standardní jádro . . . 10
4.2.2 Jádro pro tlakové pole . . . 10
4.2.3 Jádro pro viskozitu . . . 11

5 Lagrangeovská metoda **13**
5.1 Hustota . . . 14
5.2 Vnitřní síly . . . 14
5.2.1 Tlak . . . 14
5.2.2 Viskozita . . . 16
5.3 Vnější síly . . . 17
5.3.1 Gravitace . . . 17
5.3.2 Povrchové napětí . . . 17
5.4 Kolize . . . 18
5.4.1 Detekce kolize . . . 18
5.4.1.1 Rovina . . . 19
5.4.1.2 Kvádr . . . 20
5.4.1.3 Výšková mapa . . . 20
5.4.2 Reakce na kolizi . . . 21
5.5 Integrace . . . 21

6 Návrh řešení **23**

7 Popis implementace **25**
7.1 Flow . . . 25
7.2 Simulation . . . 25
7.2.1 Hledání sousedních částic . . . 25
7.2.2 Data částice . . . 26
7.2.3 Výšková mapa . . . 28
7.2.4 Rozhraní mezi simulací a vizualizací . . . 28
7.2.5 Určení vyhlazovací délky . . . 29
7.2.6 Volba tuhosti a viskozity kapaliny . . . 29
7.3 Renderer . . . 30

7.3.1 Příkazy a parametry 30
7.3.2 Vizualizace povrchu 30
7.3.3 Vizualizace stavové veličiny 31
7.3.4 Vizualizace výškové mapy 33

8 Testy a výsledky **34**
8.1 Konstanta tuhosti k 34
8.2 Konstanta viskozity μ 35
8.3 Chování na hranicích 36
8.4 Časová a paměťová složitost 37

9 Závěr **43**
9.1 Budoucí práce 43

10 Literatura **45**

A Ovládání aplikace **47**

B Struktura konfiguračního souboru **48**

C Seznam parametrů a příkazů **49**

D Obsah přiloženého CD **52**

Seznam obrázků

2.1 Losasso a spol - Simulating Water and Smoke with an Octree Data Structure. . 3
2.2 Goktekin a spol. - A Method for Animating Viscoelastic Fluids 3
2.3 Selle a spol. - A Vortex Particle Method for Smoke, Water and Explosions. . . 4
2.4 Muller a spol. - Particle-Based Fluid-Fluid Interaction. 4
2.5 Premože a spol. - Particle-Based Simulation of Fluids. 5

3.1 Eulerova metoda. 7

4.1 Izotropní Gaussovo jádro. 9
4.2 Standardní jádro, průmět do jedné dimenze pro $h = 1$. 10
4.3 Jádro pro tlakové pole, průmět do jedné dimenze pro $h = 1$. 11
4.4 Jádro pro viskozitu, průmět do jedné dimenze pro $h = 1$. 12

5.1 Lagrangeova metoda. 13
5.2 Výsledná síla způsobená nedostatkem částic v okolí. 14
5.3 Výsledná síla způsobená přebytkem částic v okolí. 15
5.4 Výsledná síla způsobená viskozitou. 16
5.5 Vznik povrchového napětí následkem nevyvážených mezimolekulárních sil. . . . 17
5.6 Diagram obecné kolize. 19
5.7 Diagram kolize s rovinou. 19
5.8 Diagram kolize s kvádrem. 2D řez. 20
5.9 Diagram kolize s výškovou mapou. 2D řez. 21
5.10 Činnost leap-frog integrátoru. 22

6.1 Návrh řešení. 23
6.2 Diagram průběhu simulace. 24

7.1 Buňky pomocné mřížky pro urychlení vyhledávání sousedních částic. 26
7.2 Vzorkování funkcí `GetPointInfo`. 28
7.3 Volba velikosti vyhlazovací délky jádra. 29
7.4 Vliv parametru *r_mc_subdiv* na kvalitu výsledného povrchu. 31
7.5 Různé druhy vizualizací. 32

8.1 Testovací scéna pro znázornění vlivu konstanty tuhosti. 34
8.2 Testovací scéna pro znázornění vlivu viskozity. 35
8.3 Diagram chování kapaliny na hranici. 36
8.4 Testovací scéna pro znázornění chování na hranicích. 36
8.5 Časové závislosti. Absolutní časy. 39
8.6 Časové závislosti. Časy pro posledních 100 iterací. 40
8.7 Alokovaná paměť. 41
8.8 Testovací scéna pro měření časové složitosti. 42

Seznam tabulek

7.1 Časy potřebné pro vzorkování a běh algoritmu pochodující krychle pro různou jemnost mřížky. 31

8.1 Parametry použité při simulacích pro zjištění časové a paměťové složitosti. n je počet částic, m hmotnost jedné částice, h je velikost okolí a k je konstanta tuhosti. 37
8.2 Průměrný počet sousedů. 37

1 Úvod

V každodenním životě přicházíme do styku s nejrůznějšími kapalinami a jsme zvyklí na jejich chování. Umíme si představit, jak se chová a vypadá voda v řece, co se stane, když zamícháme čaj v hrnku atd. Proto je překvapivé, jak složité je tyto každodenní jevy simulovat a vizualizovat. S rostoucím výkonem výpočetní techniky je možné zahrnout více aspektů chování kapalin, ale zatím není možné vytvořit univerzální model postihující všechny aspekty chování kapalin.
Modely používané ve výpočetní grafice mají své kořeny ve výsledcích výzkumu oblasti nazývané výpočetní dynamika tekutin (CFD[1]). Tento výzkum je veden potřebami inženýrských aplikací a simulací, nehledí se zde na rychlost simulace, ale především na její přesnost. Modely pro potřeby počítačové grafiky mají opačné priority a je přijatelné dopouštět se nepřesností v zájmu urychlení výpočtu. I bez požadavku vysoké přesnosti je stále obtížné simulovat kapaliny v reálném čase na úrovni vyhovující dnešním potřebám počítačové grafiky.
Základní rovnice pro modelování kapalin jsou Navier-Stokesovy rovnice (více v kapitole 3.1). Pro jejich řešení se používají dvě rozdílné metody. Eulerova, často založená na pravidelné mřížce (kapitola 3.2) a Lagrangeova, často založena na částicích (kapitola 5).
Nejčastěji jsou používané offline simulátory. Čas potřebný na výpočet jednoho snímku se u nich dnes pohybuje v jednotkách minut. Pro tyto aplikace byly s úspěchem použity modely založené na pravidelné mřížce a vizualizační postupy založené například na metodě sledování paprsku. Tyto modely se ovšem nehodí pro simulaci v reálném čase pro malé scény, ve kterých chceme zachytit drobné detaily povrchu. Pro tyto případy se v posledních letech prosazují modely založené na částicích.
Aplikace těchto modelů umožňuje simulovat kapaliny (vodu, olej, krev...), viskoelastické materiály (gely), materiály složené z drobných částic (písek, bahno), dokonce i fázové přechody (například tok a postupné tuhnutí lávy). Tyto simulace mohou být použity v různých odvětvích počínaje speciálními efekty ve filmech přes výcvikové simulátory až po herní průmysl. Existují implementace využívající speciální hardware (například fyzikální akcelerátory firmy Ageia[1]), nebo využívající výkon moderních grafických karet[3]. Při použití grafické karty přímo k simulaci odpadá nutnost přenosu dat pro vizualizaci z hlavní paměti počítače do paměti grafické karty. Tyto implementace se zaměřují na výpočet v reálném čase zejména pro potřeby herního průmyslu.

1.1 Cíl práce

Cílem mojí práce je implementace SPH[2] modelu spolu s vytvořením prostředí dovolujícího vizualizaci stavových veličin modelu a extrakci povrchu kapaliny. Prostředí poskytuje možnost ovlivnit některé parametry simulované kapaliny. Po implementaci následuje posouzení výhod a nevýhod této metody vzhledem k ostatním metodám.

1.2 Členění práce

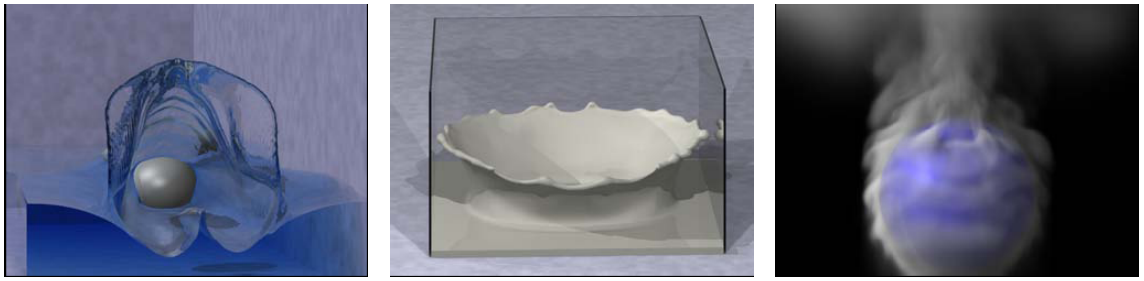
V kapitole 2 jsou v krátkosti předvedeny výsledky existujících modelů používaných v počítačové grafice. V kapitole 3 jsou předvedeny základní rovnice, ze kterých vycházejí modely simulující kapaliny, a Eulerova metoda jejich řešení. V kapitole 4 je podrobně popsána metoda SPH a její modifikace použité v této práci. Kapitola 5 popisuje řešení Lagrangeovou metodou za pomoci SPH. V kapitole 6 je představen návrh řešení, implementace tohoto návrhu je popsána v kapitole 7. Vlivy nejdůležitějších parametrů kapalin na její chování, časová a paměťová složitost jsou

[1] Computational fluid dynamics
[2] Smoothed Particle Hydrodynamics

popsány v kapitole 8. Shrnutí dosažených výsledků a možná další vylepšení modelu i aplikace jsou popsány v kapitole 9.

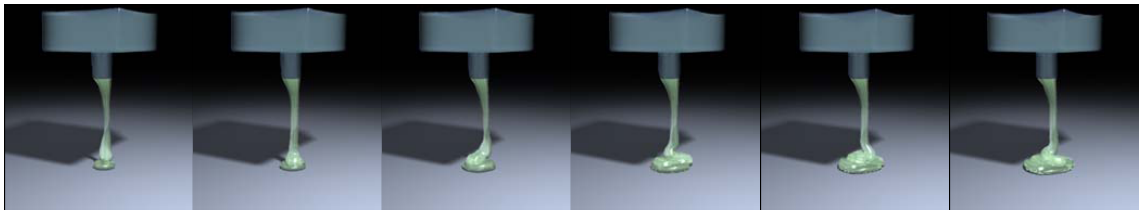
2 Stav oboru

V této kapitole budou uvedeny některé práce a jejich výsledky spolu s hrubým popisem metody, kterou používají. Začneme několika příklady modelů založených na Eulerově metodě.



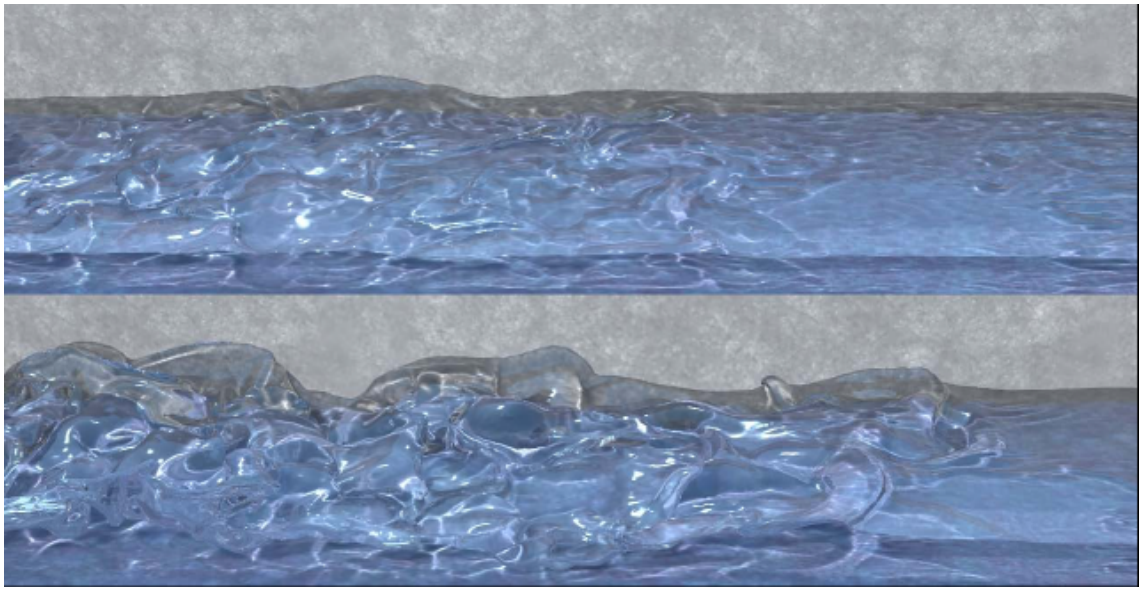
Obrázek 2.1: Losasso a spol - Simulating Water and Smoke with an Octree Data Structure [5]. Elipsa procházející mělkou nádrží vody (vlevo), vytvoření koruny na mléku (uprostřed) a kouř stoupající okolo koule (vpravo)

Jedním z hlavních problémů Eulerovy metody je paměťová náročnost 3D mřížky, ve které výpočet probíhá. V práci [5] je představena metoda, kde je pravidelná mřížka nahrazena neomezeným oktantovým stromem. Tato struktura umožní dostatečně zjemnit dělení prostoru na potřebných místech a zachytit drobné detaily. Na obrázku 2.1 jsou ukázky výsledků. Na levém snímku je dobře patrné, že metoda velmi dobře zvládá i tenké vrstvy kapaliny. Rychlost je díky použití oktantového stromu na velmi slušné úrovni. Pro efektivní rozlišení mřížky 512^3 je čas potřebný na výpočet jednoho snímku 4-5 minut. Je možné zvýšit rozlišení až na 1024^3, což umožní zachytit i drobné detaily.



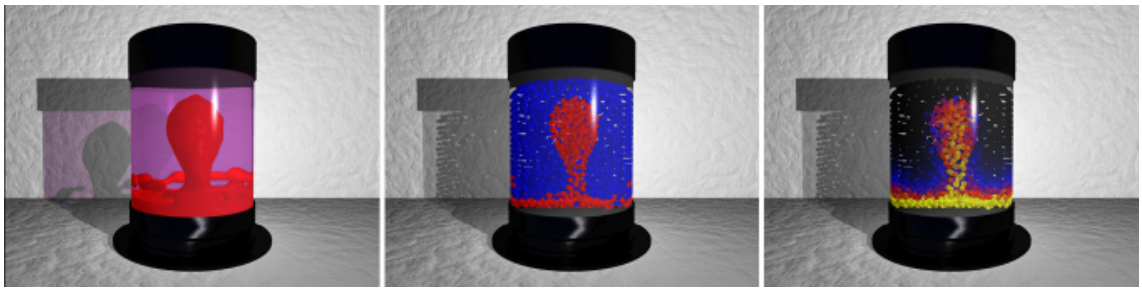
Obrázek 2.2: Goktekin a spol. - A Method for Animating Viscoelastic Fluids [2]. Výtok viskoelastické kapaliny z nádrže, proud se spirálově stáčí, což je typické pro viskoelastické kapaliny.

Ke klasickým Navier-Stokesovým rovnicím, ze kterých modely vycházejí, je možné přidat další rovnice, které modelují aspekty specifických kapalin. V práci [2] je tímto aspektem elasticita. Viskoelastické kapaliny mají vlastnosti, které spojují pevné látky a kapaliny. Tyto látky na malá napětí reagují elasticky, podobně jako pevné látky, ale pro větší napětí tečou jako kapaliny. Příkladem takových kapalin jsou různé gely, sliz, jíl, těsto a další. Na obrázku 2.2 je výtok takovéto kapaliny z nádrže. Metoda, kterou bylo toto viskoelastické chování zajištěno, je podle autorů do značné míry nezávislá na modelu kapaliny. Ačkoli jejich implementace používala Eulerovu metodu, bylo by snadné použít například SPH. Při rozlišení mřížky 40^3 bylo na vteřinu animace potřeba zhruba půl hodiny výpočetního času na 3GHz Pentium4 procesoru.
Simulace turbulentních proudění, i když nepřesná (fyzikálně správná simulace turbulentních proudění je stále nevyřešený úkol), je obtížný úkol. Bylo vyvinuto několik metod, které umožňují zachytit vírovost turbulentního proudění i při použití Eulerovy metody. Hlavní z nich je uzavření vírovosti (vorticity confinement). Tato metoda může pouze zesílit již existující vírovost a pokud



Obrázek 2.3: Selle a spol. - A Vortex Particle Method for Smoke, Water and Explosions [11]. Kapalina teče zleva doprava, na vtoku jsou generovány vírové částice, ve spodní scéně mají částice více vírovosti.

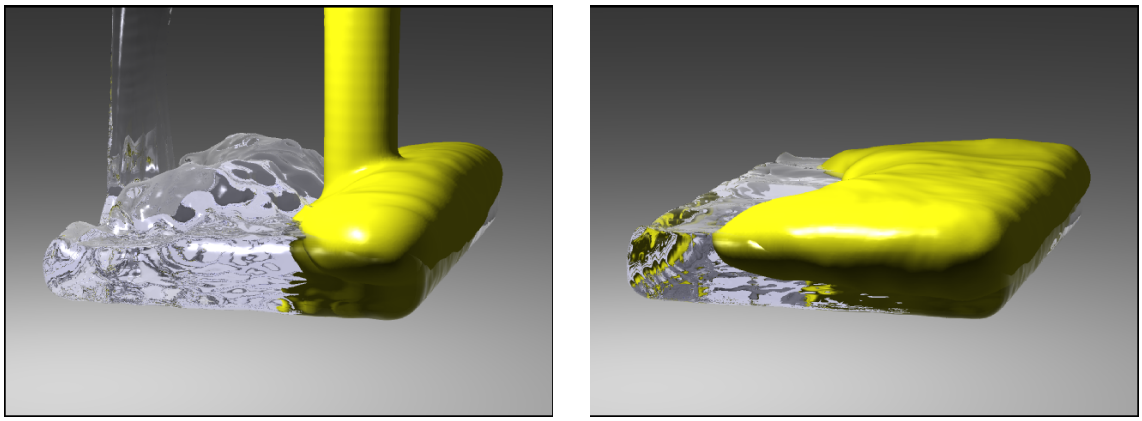
není mřížka dostatečně jemná, vírovost nebude zachycena. V práci [11] autoři spojily výhody Eulerovy metody s výhodami Lagrangeovy metody použitím vírových částic uvnitř Eulerovy mřížky. Na obrázku 2.3 je vidět výsledek tohoto spojení. Vliv částic na rychlost simulace je minimální (méně než 5%). Každá částice má daný efektivní poloměr, ve kterém ovlivňuje buňky mřížky. Zajímavé efekty se dají dosáhnout použitím různých poloměrů, přičemž malé poloměry zastupují drobné turbulence a velké poloměry odpovídají velkým turbulencím. Na obrázku jsou použity dva poloměry-jeden odpovídající rozměru 4 buňek, druhý odpovídající rozměru 40 buněk. Částice s malým poloměrem mají též přibližně desetkrát menší vírovost.



Obrázek 2.4: Muller a spol. - Particle-Based Fluid-Fluid Interaction [8]. Lávová lampa, výsledný povrch (vlevo), částice obou kapalin (uprostřed) a teploty jednotlivých částic(vpravo)

Druhým přístupem k simulaci kapalin je Lagrangeova metoda. Mezi výhody této metody patří například schopnost popsání interakce mezi více kapalinami, či mezi kapalinou a plynem, což je při použití Eulerovy metody nadmíru obtížné. V práci [8] je použita metoda SPH, která byla rozšířena tak, aby byla schopna modelovat více druhů kapalin, plyn ve formě bublin v kapalině a var kapaliny. Na obrázku 2.4 je lávová lampa, která ukazuje interakci mezi dvěmi nemísitelnými kapalinami, závislost hustoty na teplotě a také vedení tepla. Autoři také úspěšně simulovali var vody a zachycení vzduchových kapes při nalévání tekutiny do kontejneru. Vlastní simulace

používající několik tisíc částic je prováděna interaktivní rychlostí, ale hledání povrchu a jeho vykreslení je značně pomalé. Další práce využívající SPH animuje tok lávy [12].



Obrázek 2.5: Premože a spol. - Particle-Based Simulation of Fluids [10]. Míchaní dvou nemísitelných kaplin.

Poměrně novou metodou pracující s částicemi je MPS[1], která byla vyvinuta pro řešení Navier-Stokesových rovnic pro nestlačitelné kapaliny. Tato metoda je schopna řešit široký okruh jevů včetně fázového přechodu, vícefázový tok, elastické chování, tok více různých kapalin a další. V práci [10] autoři implementovali tuto metodu s důrazem na snadné řízení simulace ze strany uživatele. Tato metoda je podle autorů vhodná i na simulace větších rozměrů, jelikož je založena na částicích, které jsou paměťově i výpočetně úspornější než mřížkové metody. Na obrázku 2.5 je simulace mísení dvou kapalin. Pro tuto simulaci bylo použito 80000 částic a simulace zabrala 3 minuty. Problémem se opět ukázalo získání povrchu, které trvalo plných 10 minut. Autoři také simulovali zaplavení chodby. Použili 100000 částic při simulačním čase přibližně 3 minuty. Dosáhli výsledků srovnatelných s metodami založenými na mřížce, které by ovšem nebyly schopny poskytnout takové množství detailů v tak velkém prostoru při splnitelných paměťových a časových nárocích.

[1] Moving Particle Semi-Implicit

3 Klasická výpočetní hydrodynamika

3.1 Navier-Stokesovy rovnice

Navier-Stokesovy rovnice[15] jsou základní rovnice popisující pohyb tekutin. Jedná se o sadu diferenciálních rovnic, které pro většinu reálných situací přecházejí v nelineární parciální diferenciální rovnice. Jejich řešení je značně obtížné. Těmito rovnicemi jsme schopni popsat mnoho typů proudění. Jakmile se začne projevovat vliv turbulence, analytické řešení zatím není známo a numerické metody nejsou schopny zachytit potřebné detaily, aniž by řešení zabralo příliš mnoho času. Pro některé případy (například proudění vzduchu kolem křídla) byly empiricky zjištěny modely umožňující dostatečně přesné výpočty.
V této práci se budeme zabývat prouděním nestlačitelných newtonských kapalin. Rysem newtonské kapaliny je, že dynamická viskozita je konstanta úměrnosti mezi napětím a rychlostí deformace.

$$\tau = -\eta \frac{du}{dt} \tag{3.1}$$

τ je tečné napětí v tekutině, u je rychlost toku, x je souřadnice ve směru kolmém na směr proudění a η je dynamická viskozita. Pro tyto kapaliny mají Navier-Stokesovy rovnice následující tvar

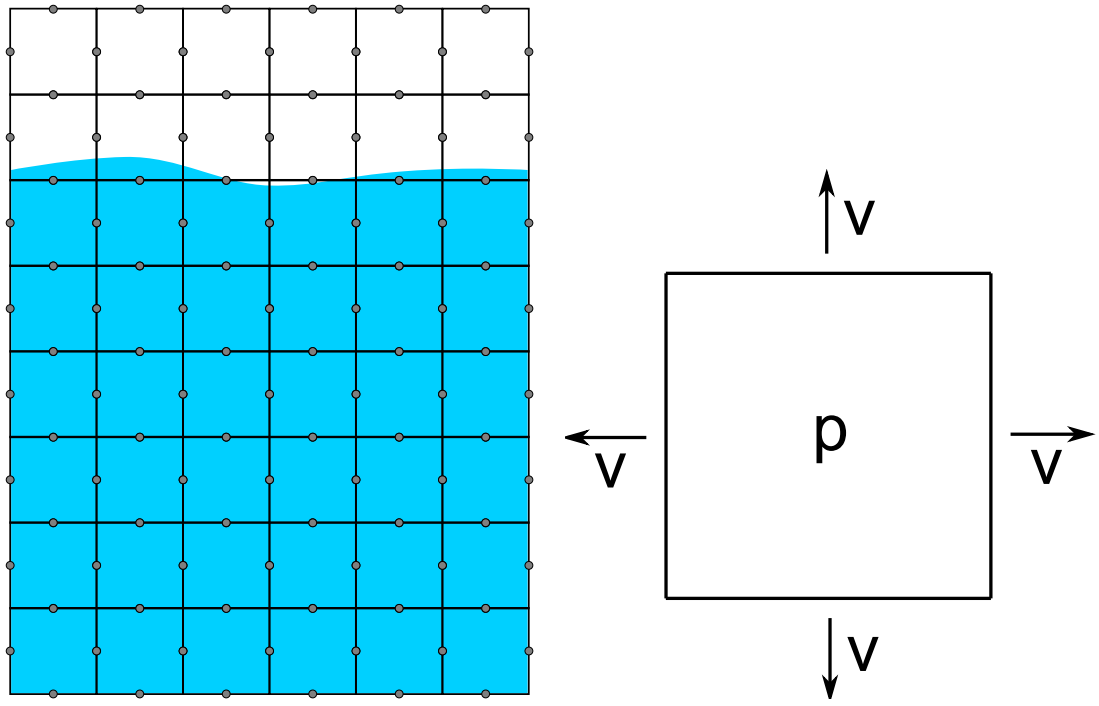
$$\rho(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v}) = -\nabla \mathbf{p} + \mu \nabla^{\mathbf{2}} \mathbf{v} + \mathbf{f}, \tag{3.2}$$

$$\nabla \cdot \mathbf{v} = 0, \tag{3.3}$$

kde $\mathbf{v}$ je vektor rychlosti, ρ hustota kapaliny, p tlak, μ viskozita a $\mathbf{f}$ vektor externích sil (například gravitace).
Rovnice 3.2 popisuje vztah mezi setrvačností, gradientem tlakového pole, viskozitou a externími silami. Jedná se vlastně o zákon zachování energie pro kapaliny. Na pravé straně rovnice jsou síly (tlaková, viskozní, externí) a na straně levé součin hustoty a zrychlení. Zrychlení je složeno ze dvou částí - nerovnoměrného $\frac{\partial \mathbf{v}}{\partial t}$ závisejícího na čase a konvektivního $\mathbf{v} \cdot \nabla \mathbf{v}$ závisejícího na pozici.
Rovnice 3.3 je přepsaný zákon zachování objemu (hmoty) pro nestlačitelné kapaliny.

3.2 Eulerova Metoda

Klasická Eulerova metoda pracuje s představou kapaliny rozdělené do buněk (cells) vyrovnaných do pravidelné mřížky. Na obrázku 3.1 je znázorněno 2D schéma rozdělení prostoru do buněk. Detail buňky je na obrázku 3.1, na kterém je vidět, že rychlosti jsou uchovávány pro každou stěnu a jsou tudíž sdíleny sousedními buňkami, zatímco tlak je uchováván v buňce. Tato metoda vychází přímo z Navier-Stokesových rovnic, je tedy používána nejdéle a dnes je již velmi dobře zvládnuta.
Ačkoli některé stavové proměnné jsou touto metodou dobře popsány, hlavním omezením této metody je vlastní mřížka. Kapalina nemůže existovat mimo mřížku. Při nutnosti pokrýt rozsáhlejší scénu mřížkou dostatečně jemnou pro zachycení detailů jsou paměťové nároky i výpočetní složitost velmi vysoké. Novější implementace se opírají o modernější datové struktury, které jsou v průběhu simulace rozšiřitelné, a je možno v blízkosti zajímavých míst provést rozdělení na jemnější buňky. Příkladem takové struktury je oktantový strom použitý v [5]. Dále jsou dnes používány hybridní přístupy, v nichž je část problému řešena pomocí mřížek a část pomocí částic, všechny tyto metody ale ztěžují implementaci a/nebo často vedou ke specializaci na určitý druh kapalin. I přes použití takovýchto postupů se metody založené na mřížce nehodí pro interaktivní simulaci detailů povrchu kapalin a mají svoje zastoupení spíše ve filmovém průmyslu.



Obrázek 3.1: Eulerova metoda, diskrétní pole rychlostí je znázorněno body. Na pravém obrázku je detail jedné buňky.

4 Smoothed particle hydrodynamics

Podle [13] byla metoda SPH vyvinuta v roce 1977 pro potřeby simulace toku mezihvězdného plynu v astrofyzice. Jde tedy o model proudění stlačitelné tekutiny. SPH je interpolační metoda používající částice. Každá částice je nositelkou určitých entit, jako jsou hmotnost, tlak, pozice, rychlost atd. Příspěvky od jednotlivých částic v daném bodě jsou váženy *jádrem* (kernel function), vzdálenější částice mají menší vliv než částice bližší. Toto jádro pracuje jako filtr, který rozprostře entity částice z diskrétního bodu do jejího okolí. Velkou výhodou této metody je, absence mřížky a také dobrá kontrola nad rozlišením výsledné simulace.

4.1 Definice

Interpolace použitá v SPH je založena na teorii integrální interpolace. Integrální interpolace funkce $A(\mathbf{r})$ je definována v celém prostoru Ω vztahem

$$A_I(\mathbf{r}) = \int_\Omega A(\mathbf{r}) W(\mathbf{r} - \mathbf{r}', h)\, d\mathbf{r}', \tag{4.1}$$

kde $\mathbf{r}$ je bod kde vyhodnocujeme funkci A, W je jádro, h je parametr jádra určující vyhlazující délku (smoothing length) a $\mathbf{r}'$ je nová nezávislá proměnná. Podle [14] jsou na jádro kladeny jisté požadavky, musí být normalizované, tedy

$$\int W(\mathbf{r} - \mathbf{r}', h)\, d\mathbf{r}' = 1$$

pro h blížící se nule se funkce redukuje na Diracovu funkci delta

$$\lim_{h \to 0} W(\mathbf{r} - \mathbf{r}', h) = \delta(\mathbf{r} - \mathbf{r}'),$$

z čehož plyne

$$\lim_{h \to 0} A_I(\mathbf{r}) = A(\mathbf{r}).$$

Pokud známe funkci $A(\mathbf{r})$ pouze v N diskrétních bodech, můžeme integrál z rovnice 4.1 aproximovat sumou

$$A_I(\mathbf{r}) = \sum_{j=1}^{N} V_j A(\mathbf{r}_j) W(\mathbf{r} - \mathbf{r}_j, h), \tag{4.2}$$

kde V_j je objem asociovaný s částicí j, $\mathbf{r}_j$ je pozice částice j. Objem V_j typicky vyjádříme z hustoty a hmotnosti přiřazené částici

$$V_j = \frac{m_j}{\rho_j}$$

a rovnici 4.2 přepíšeme do tvaru

$$A_I(\mathbf{r}) = \sum_{j=1}^{N} \frac{m_j}{\rho_j} A(\mathbf{r}_j) W(\mathbf{r} - \mathbf{r}_j, h). \tag{4.3}$$

Tento tvar bude výchozím bodem pro implementaci Lagrangeovy metody.
Ve většině rovnic se objevují derivace a je proto třeba definovat jak se v SPH tyto derivace počítají. Zde se projeví jedna z hlavních vlastností SPH, derivace se totiž ve výsledku týká pouze jádra. Zderivujeme-li obě strany rovnice

$$\nabla A_I(\mathbf{r}) = \nabla \sum_{j=1}^{N} \frac{m_j}{\rho_j} A(\mathbf{r}_j) W(\mathbf{r} - \mathbf{r}_j, h),$$

jediné co na pravé straně rovnice závisí na $\mathbf{r}$ je jádro W, po úpravě tedy získáme tvar

$$\nabla A_I(\mathbf{r}) = \sum_{j=1}^{N} \frac{m_j}{\rho_j} A(\mathbf{r}_j) \nabla W(\mathbf{r} - \mathbf{r}_j, h). \tag{4.4}$$

Tuto vlastnost lze analogicky odvodit pro laplacián

$$\nabla^2 A_I(\mathbf{r}) = \sum_{j=1}^{N} \frac{m_j}{\rho_j} A(\mathbf{r}_j) \nabla^2 W(\mathbf{r} - \mathbf{r}_j, h). \tag{4.5}$$

4.2 Jádra

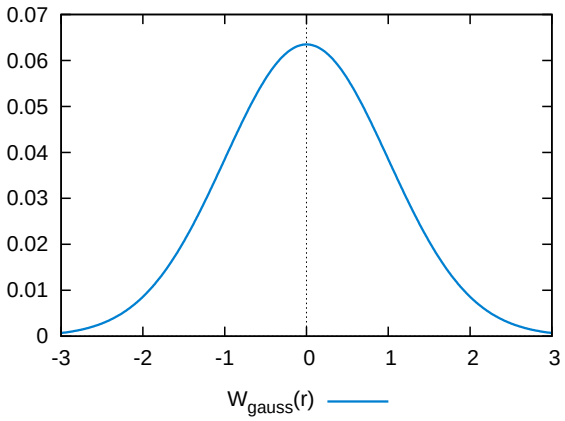
V předchozí kapitole jsme se seznámili s pojmem jádra a zformulovali několik podmínek, které jsou na tuto funkci kladeny. Další podmínky nejsou potřebné z hlediska odvození SPH, ale pro dobrou funkci. Patří mezi ně nezápornost jádra v celém definičním oboru

$$W(\mathbf{r}, h) >= 0,$$

sudost, protože očekáváme izotropní chování

$$W(\mathbf{r}, h) = W(-\mathbf{r}, h).$$

Tyto vlastnosti splňuje a v teoretických aplikacích se dle [4] používá izotropní Gaussovo jádro,



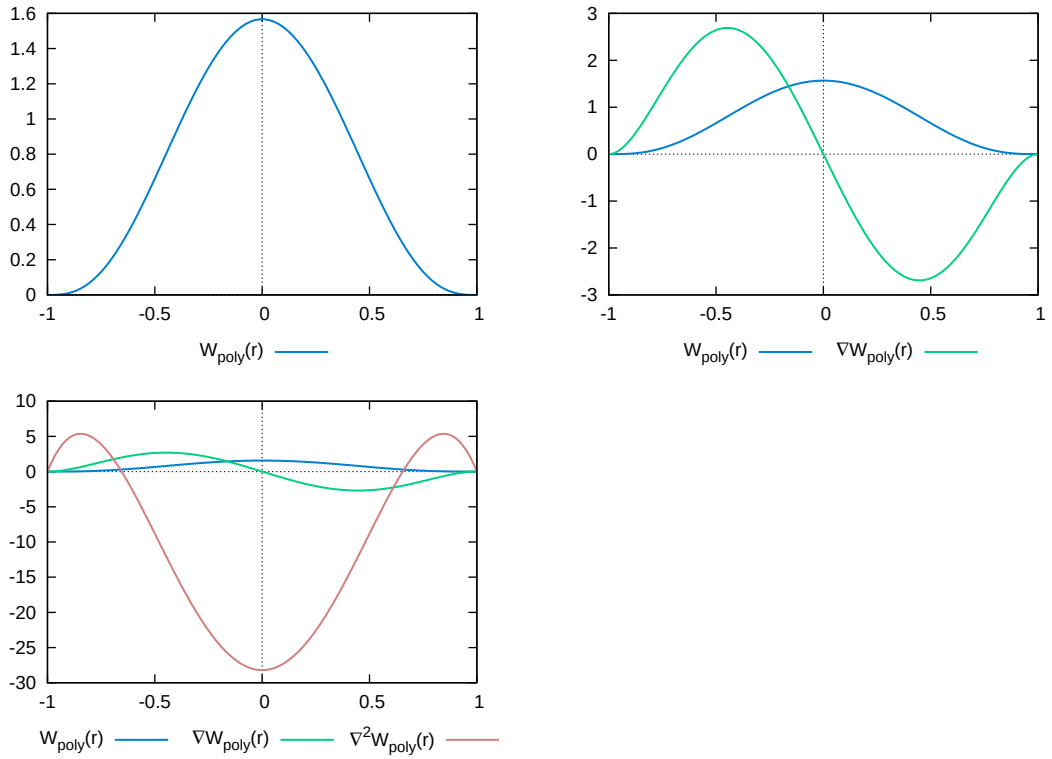
Obrázek 4.1: Izotropní Gaussovo jádro.

$$W_{gauss}(\mathbf{r}, h) = \frac{1}{(2\pi h^2)^{\frac{3}{2}}} \exp^{-\frac{|r|^2}{2h^2}}, \quad h > 0,$$

jehož průběh je znázorněn na obrázku 4.1. Toto jádro má vhodné matematické vlastnosti, ale pro praktický výpočet se nehodí. Za prvé je pro výpočet potřeba vyhodnotit exponenciální funkci a za druhé není omezené, takže jakákoliv částice přispívá k výsledku, byť zanedbatelně. Proto se v praxi používají jádra omezená, takže poslední podmínka na jádro je

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

V následujících podkapitolách jsou uvedena použitá jádra, jejich zápisy a grafy průběhů.



Obrázek 4.2: Standardní jádro, průmět do jedné dimenze pro $h = 1$.

4.2.1 Standardní jádro

Standardní jádro (obrázek 4.2) je použito pro výpočet všech veličin, pokud není specifický důvod použít jádro jiné. Toto jádro má podobný tvar jako jádro Gaussovo, ale jeho vyhodnocení je rychlé, neobsahuje totiž žádnou složitou funkci ani odmocninu, vzdálenost částice od bodu je potřeba v druhé mocnině. Výpočet odmocniny je časově velmi náročný a je vhodné se mu vyhnout, je-li to možné. Základní tvar jádra je

$$W_{poly}(\mathbf{r}, h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - |\mathbf{r}|^2)^3 & , \quad 0 \leq |\mathbf{r}| \leq h \\ 0 & , \quad jinak \end{cases}, \tag{4.6}$$

gradientu

$$\nabla W_{poly}(\mathbf{r}, h) = -\frac{945}{32\pi h^9}(h^2 - |\mathbf{r}|^2)^2\mathbf{r}, \quad 0 \leq |\mathbf{r}| \leq h, \tag{4.7}$$

a laplaciánu

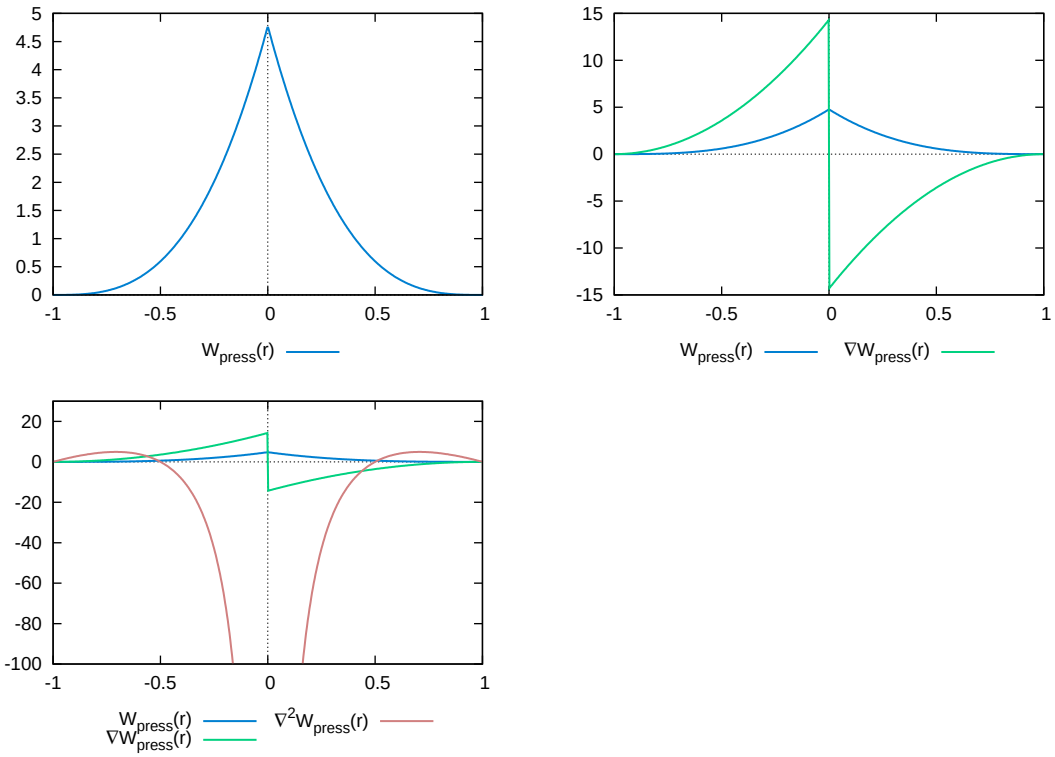
$$\nabla^2 W_{poly}(\mathbf{r}, h) = -\frac{945}{32\pi h^9}(h^2 - |\mathbf{r}|^2)(3h^2 - 7|\mathbf{r}|^2), \quad 0 \leq |\mathbf{r}| \leq h. \tag{4.8}$$

V rovnicích jsou odděleny části, které lze předpočítat, a části, které jsou závislé na $\mathbf{r}$.

4.2.2 Jádro pro tlakové pole

Z průběhu na obrázku 4.2 je zřejmé, že derivace je nulová pro hodnoty $\mathbf{r} \to 0$, což je pro výpočet tlakového pole (kapitola 5.2.1) nepoužitelné, a je tedy potřeba použít jádro s nenulovou derivací v počátku. Jedno z používaných jader je toto (obrázek 4.3)

$$W_{press}(\mathbf{r}, h) = \frac{15}{\pi h^6} \begin{cases} (h - |\mathbf{r}|)^3 & , \quad 0 \leq |\mathbf{r}| \leq h \\ 0 & , \quad jinak \end{cases}, \tag{4.9}$$



Obrázek 4.3: Jádro pro tlakové pole, průmět do jedné dimenze pro $h = 1$.

s gradientem

$$\nabla W_{press}(\mathbf{r}, h) = -\frac{45}{\pi h^6} \frac{\mathbf{r}}{|\mathbf{r}|}(h - |\mathbf{r}|)^2,$$
$$\lim_{r \to 0^-} \nabla W_{press}(r, h) = \frac{45}{\pi h^6}, \quad \lim_{r \to 0^+} \nabla W_{press}(r, h) = -\frac{45}{\pi h^6} \tag{4.10}$$

a laplaciánem

$$\nabla^2 W_{press}(\mathbf{r}, h) = -\frac{90}{\pi h^6} \frac{1}{|\mathbf{r}|}(h - |\mathbf{r}|)(h - 2|\mathbf{r}|),$$
$$\lim_{r \to 0} \nabla^2 W_{press}(r, h) = -\infty. \tag{4.11}$$

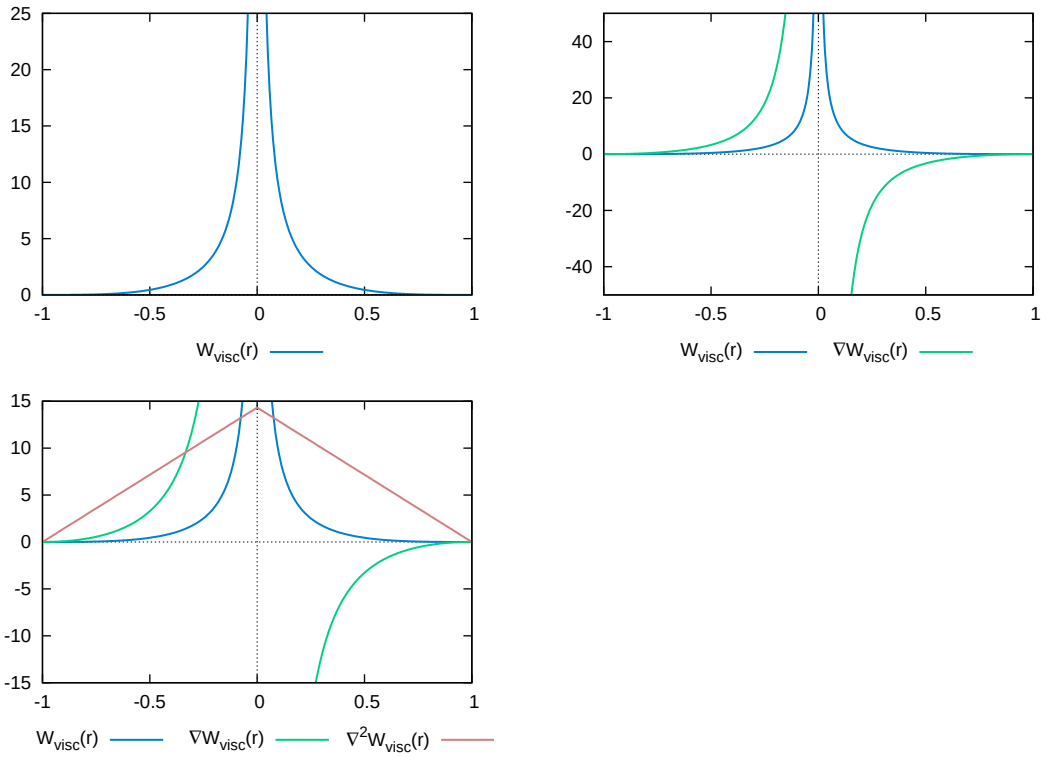
4.2.3 Jádro pro viskozitu

Pro výpočet vnitřních sil odpovídajících viskozitě, je třeba laplacián jádra (kapitola 5.2.2) a obě předchozí jádra mají pro část, nebo celý definiční obor záporný laplacián. Proto potřebujeme speciální jádro s kladným laplaciánem. Používá se toto jádro (obrázek 4.4)

$$W_{visc}(\mathbf{r}, h) = \frac{15}{2\pi h^3} \begin{cases} \frac{|\mathbf{r}|^3}{2h^3} + \frac{|\mathbf{r}|^2}{h^2} + \frac{h}{2|\mathbf{r}|} - 1 & , \quad 0 \le |\mathbf{r}| \le h \\ 0 & , \quad jinak \end{cases},$$
$$\lim_{r \to 0} W_{visc}(r, h) = \infty, \tag{4.12}$$

s gradientem

$$\nabla W_{visc}(\mathbf{r}, h) = \frac{15}{2\pi h^3} \mathbf{r}\left(-\frac{3|\mathbf{r}|}{2h^3} + \frac{2}{h^2} - \frac{h}{2|\mathbf{r}|^3}\right),$$
$$\lim_{r \to 0^-} \nabla W_{visc}(r, h) = +\infty, \quad \lim_{r \to 0^+} \nabla W_{visc}(r, h) = -\infty, \tag{4.13}$$

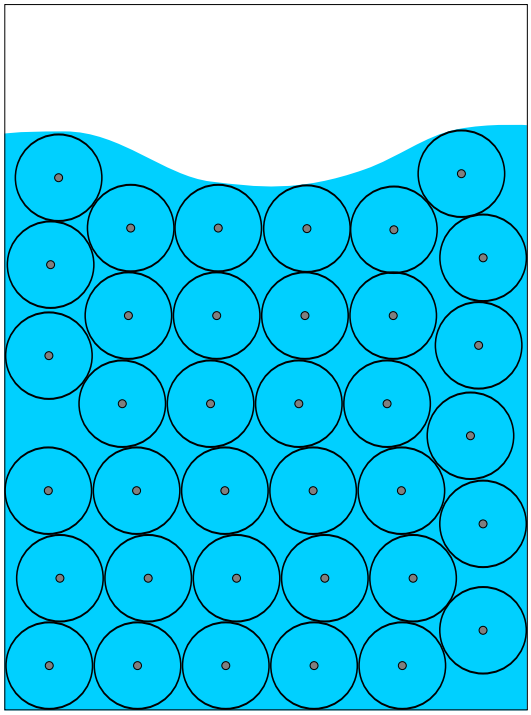


Obrázek 4.4: Jádro pro viskozitu, průmět do jedné dimenze pro $h = 1$.

a laplaciánem

$$\nabla^2 W_{visc}(\mathbf{r}, h) = \frac{45}{\pi h^6}(h - |\mathbf{r}|)). \tag{4.14}$$

5 Lagrangeovská metoda



Obrázek 5.1: Lagrangeova metoda, diskrétní pole rychlostí je znázorněno body uvnitř částic.

Zatímco Eulerova metoda (kapitola 3.2) pracuje s kapalinou rozdělenou pravidelnou mřížkou, Lagrangeova metoda rozděluje kapalinu mezi množství částic. Tyto částice typicky nejsou nijak propojeny. Existují i Lagrangeovy metody s mřížkou, jejich použití ale není rozšířené kvůli velké deformaci mřížky, budeme se proto dále zabývat pouze bezmřížkovou metodou. Každá částice reprezentuje určité množství kapaliny, má tedy danou hmotnost, pozici, rychlost a další veličiny používané pro výpočet. Na obrázku 5.1 je znázorněn 2D příklad Lagrangeova popisu kapaliny.
Navier-Stokesovy rovnice (3.2) a (3.3) se při užití Lagrangeovy metody zjednoduší. Rovnice (3.3) je zákon zachování hmoty pro kapaliny. Budeme-li předpokládat konstantní počet částic a konstantní hmotnost jedné částice, pak je zachování hmoty zaručeno a není třeba tuto rovnici uvažovat. Také rovnice (3.2) se zjednoduší, neboť zrychlení částice je derivace pozice částice v čase (částice se pohybuje s kapalinou, proto není potřeba konvektivní akcelerace). Základní formulace Navier-Stokesoy rovnice pro Lagrangeovu metodu je

$$\rho\frac{D\mathbf{v}}{Dt} = -\nabla p + \mu\nabla^2\mathbf{v} + \mathbf{f}. \tag{5.1}$$

Na pravé straně rovnice, jsou tři členy, odpovídající třem silovým polím. $-\nabla p$ modelující tlakové pole, $\mu\nabla^2\mathbf{v}$ modelující viskozitu a $\mathbf{f}$ odpovídající vnějším silám. Součet těchto tří silových polí $\mathbf{F} = -\nabla p + \mu\nabla^2\mathbf{v} + \mathbf{f}$ určuje změnu hybnosti částice $\rho\frac{D\mathbf{v}}{Dt}$, a tedy akcelerace částice i je

$$\mathbf{a}_i = \frac{\partial\mathbf{v}}{\partial t} = \frac{\mathbf{F}_i}{\rho_i}. \tag{5.2}$$

V následujících podkapitolách je popsáno modelování těchto silových polí pomocí SPH.

5.1 Hustota

Abychom mohli aplikovat SPH na veličiny, jejich gradienty a laplaciány (rovnice (4.3) (4.4) a (4.5)), potřebujeme znát hmotnost částice m_i, která je konstantou pro částici, a hustotu ρ_i, která je spojitou veličinou pro kapalinu, a již je třeba vypočítat. Aplikací rovnice (4.3) dojdeme k výrazu

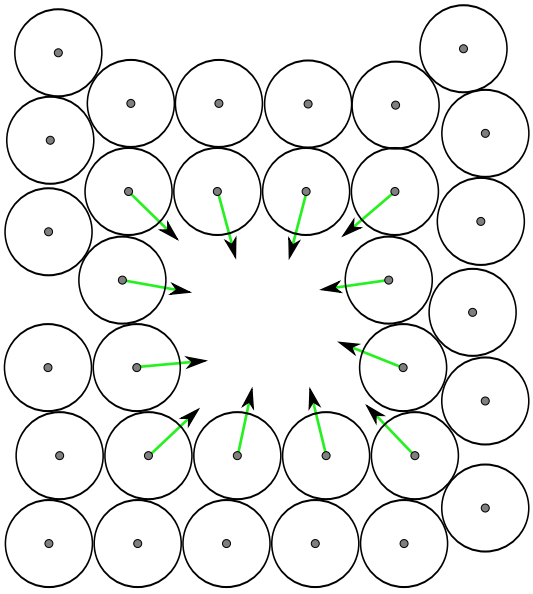
$$\begin{aligned} \rho_i &= \rho(\mathbf{r}_i) \\ &= \sum_{j=1}^{N} \rho_j \frac{m_j}{\rho_j} W(\mathbf{r} - \mathbf{r}_j, h) \\ &= \sum_{j=1}^{N} m_j W(\mathbf{r} - \mathbf{r}_j, h) \end{aligned}$$

Jádro použité v této rovnici je W_{poly} popsané v kapitole 4.2.1.

5.2 Vnitřní síly

Vnitřní síly jsou takové síly, které vznikají v kapalině samé a z rovnice 5.1 jsou to síly tlakového pole a síly viskózní, které jsou prvním a druhým členem na pravé straně této rovnice.

5.2.1 Tlak



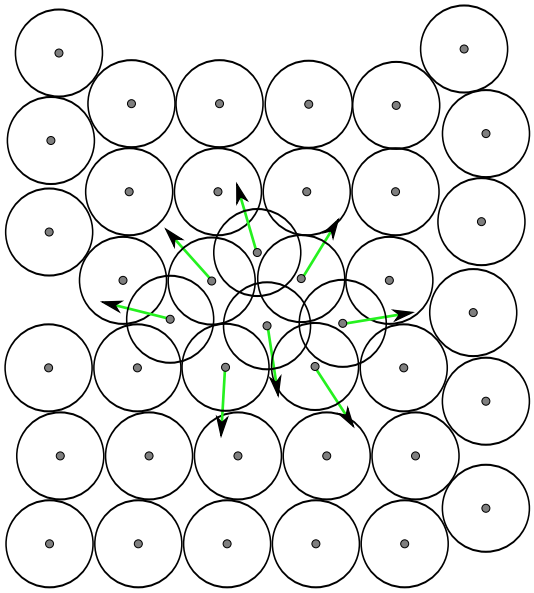
Obrázek 5.2: Výsledná síla způsobená nedostatkem částic v okolí.

První člen na pravé straně rovnice (5.1) má za následek pohyb částice ve směru opačném ke gradientu tlakového pole, částice má tendenci pohybovat se k minimu tlakového pole. Síla působící na částici je tedy

$$\mathbf{f}_i^{press} = -\nabla p(\mathbf{r}_i).$$

Aplikací SPH podle rovnice (4.4) získáme vztah

$$\mathbf{f}_i^{press} = \sum_{j=1}^{N} p_j \frac{m_j}{\rho_j} \nabla W(\mathbf{r} - \mathbf{r}_j, h). \tag{5.3}$$



Obrázek 5.3: Výsledná síla způsobená přebytkem částic v okolí.

Abychom mohli tuto sílu vyhodnotit, potřebujeme znát tlak v místě každé částice. Hodnotu odvodíme ze stavové rovnice ideálního plynu

$$pV = nRT,$$

kde p je tlak, $V = \frac{1}{\rho}$ objem na jednotku hmotnosti, n počet částic v jednom molu, R je plynová konstanta a T je teplota. Pro konstantní hmotu a teplotu je pravá strana rovnice konstantní, a proto ji nahradíme konstantou tuhosti k. Rovnici pak můžeme přepsat do tvaru

$$\begin{aligned} pV &= k \\ p\tfrac{1}{\rho} &= k \\ p &= k\rho \end{aligned}$$

Tato rovnice je v pořádku pro ideální plyn. Výsledná síla by byla z pohledu částic vždy odpuzující, což odpovídá tendenci plynu rozpínat se do celého prostoru a zaujmout stav s konstantním tlakem v celém prostoru. Pro kapaliny však potřebujeme modelovat kohezní síly a klidovou hustotu. Podle [7] použijeme modifikovaný vztah pro výpočet tlaku

$$p = k(\rho - \rho_0),$$

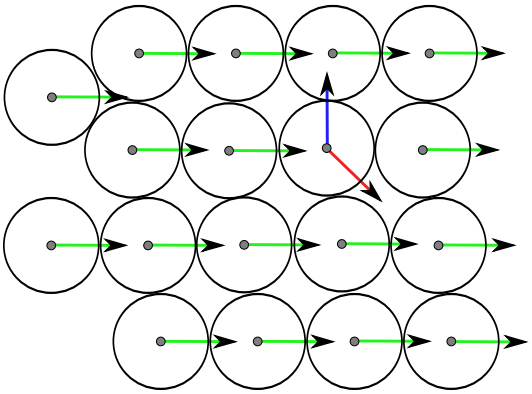
kde ρ_0 je klidová hustota. Podle tohoto vztahu může být, pro místa s hustotou menší než klidovou, hodnota tlaku záporná. Záporný tlak bude mít za následek přitažlivou tlakovou sílu. Problém rovnice (5.3) je, že neprodukuje symetrické síly. Budou-li na sebe působit dvě částice, pak výsledná síla každé z nich bude jiná, pokud hodnota tlaku v místě každé částice bude jiná. Nejjednodušší symetrizace [7] je

$$\mathbf{f}_i^{press} = \sum_{j=1}^{N} m_j \frac{p_i + p_j}{2\rho_j} \nabla W(\mathbf{r} - \mathbf{r}_j, h),$$

která je rychlá a stabilní.

Dalším problémem je volba jádra. Při použití standardního jádra, respektive jeho gradientu (4.7), dochází k nežádoucímu shlukování částic. To je následkem průběhu gradientu v okolí počátku. Pokud se částice dostanou dostatečně blízko k sobě, hodnota gradientu začne klesat a částice se neodpudí. Výsledkem jsou shluky částic, které mají vysokou hustotu. Proto je třeba použít jádro W_{press} (4.10) popsané v kapitole 4.2.2
Na obrázcích 5.2 a 5.3 jsou zjednodušeně znázorněny síly pro dvě konfigurace částic. V prvním případě existuje prostor se záporným tlakem a síla působící na částice směřuje do tohoto prostoru. V druhém případě existuje prostor s kladným tlakem (hustota je vyšší než klidová) a síla působící na částice směřuje ven z tohoto prostoru.

5.2.2 Viskozita



Obrázek 5.4: Výsledná síla způsobená viskozitou. Zelené vektory jsou rychlosti okolních částic, modrý vektor je rychlost aktuální částice a červený vektor výsledná síla.

Kapaliny nejsou schopny odolávat namáhání ve střihu a jsou-li mu vystaveny tečou. Při tečení dochází na úrovni molekul ke tření, které přeměňuje kinetickou energii v teplo. Tento odpor k tečení je označován jako viskozita a v rovnici (5.1) jí odpovídá druhý člen na pravé straně. Vnitřní síla odpovídající viskozitě je tedy

$$\mathbf{f}_i^{visc} = \mu \nabla^2 \mathbf{v},$$

kde μ je koeficient viskozity. Prostou aplikací SPH podle rovnice (4.5) získáme vztah

$$\mathbf{f}_i^{visc} = \mu \sum_{j=1}^{N} \mathbf{v}_j \frac{m_j}{\rho_j} \nabla^2 W(\mathbf{r} - \mathbf{r}_j, h).$$

Tato síla je opět nesymetrická, podle [7] viskozita závisí pouze na relativní rychlosti částic a rovnici lze přirozeně symetrizovat do tvaru

$$\mathbf{f}_i^{visc} = \mu \sum_{j=1}^{N} m_j \frac{\mathbf{v}_j - \mathbf{v}_i}{\rho_j} \nabla^2 W(\mathbf{r} - \mathbf{r}_j, h).$$

Interpretace této rovnice je pohled z hlediska pohybující se částice, tato částice bude urychlena ve směru pohybu jejího okolí. Na obrázku 5.4 je vyobrazena situace, kdy se jedna částice pohybuje jiným směrem než její okolí. Zelené vektory jsou rychlosti okolních částic, modrý vektor je rychlost aktuální částice a červený vektor je výsledná síla zapříčiněná viskozitou. Tato síla, pokud by byla jedinou silou působící na částici, by měla za následek změnu rychlosti částice takovým způsobem, aby její rychlost byla stejná jako rychlost okolí.

Při výpočtu $\mathbf{f}_i^{visc}$ je zapotřebí laplacián jádra. Standardní jádro i jádro pro tlakové pole mají laplacián v části definičního oboru kladný a v části záporný, což by vedlo k nesprávnému chování. Proto je třeba speciální jádro W_{visc} (4.14) s laplaciánem kladným v celém definičním oboru.

5.3 Vnější síly

Vnější síly jsou v rovnici 5.1 zastoupeny na pravé straně výrazem $\mathbf{f}$, který můžeme přepsat do tvaru

$$\mathbf{f}^{external} = \sum_n \mathbf{f}^n,$$

kde $\mathbf{f}^n$ odpovídá každé jednotlivé síle. Některé z externích sil mohou být aplikovány přímo na částice bez použití SPH, jiné závisí na okolních částicích. Na kolize lze také nahlížet jako na externí síly, což je detailně probráno v kapitole 5.4.

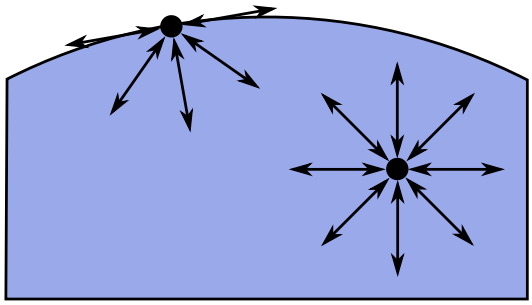
5.3.1 Gravitace

Gravitační síla působí shodně na všechny částice, je nezávislá na jejich pozici či okolí, a není proto třeba použít SPH pro její aplikaci. Pro každou částici stačí připočíst sílu

$$\mathbf{f}_i^{external} = \rho_i \mathbf{g},$$

kde $\mathbf{g}$ je gravitační zrychlení. Vememe-li v úvahu rovnici (5.2) a dosadíme-li $\mathbf{f}_i^{external}$, člen ρ_i se zkrátí, a proto můžeme gravitační zrychlení přímo přičíst ke zrychlení způsobenému silami.

5.3.2 Povrchové napětí



Obrázek 5.5: Vznik povrchového napětí následkem nevyvážených mezimolekulárních sil.

Povrchové napětí vzniká na rozhraní mezi kapalinou a okolím. Povrchové napětí je způsobeno mezimolekulárními silami, které jsou uvnitř objemu kapaliny v rovnováze a vzájemně se vyruší, ale v povrchové vrstvě mají výslednici kolmou na povrch kapaliny a směřují dovnitř objemu kapaliny. Schematické znázornění vzniku povrchového napětí je na obrázku 5.5. Toto napětí má také za následek snahu kapaliny zaujmout stav s nejmenším povrchem a nejmenší křivostí povrchu. Proto kapaliny v prostředí bez gravitace zaujímají tvar koule, protože je symetrická a z těles má nejmenší poměr povrchu ku objemu. Velikost povrchového napětí závisí na křivosti povrchu. Čím je povrch křivější, tím vyšší je povrchové napětí. Závisí také na koeficientu σ, který závisí na materiálech tvořících rozhraní.
Rovnice (5.1) neobsahuje člen popisující povrchové napětí kapaliny, a proto je třeba tento jev modelovat jako vnější sílu. Podle [7] je nutno nejprve nalézt povrch kapaliny. K tomu nám

pomůže další veličina c, kterou nazveme *obarvení* (colour field). Ta je definována hodnotou 1 na pozici částice a hodnotou 0 jinde. Po aplikaci SPH získáme vztah

$$c_i(\mathbf{r}) = \sum_{j=1}^{N} m_j \frac{1}{\rho_j} W(\mathbf{r}_i - \mathbf{r}_j, h).$$

Gradient tohoto pole, je vnitřní normálou povrchu.

$$\mathbf{n} = \nabla c$$

Po aplikaci SPH tedy získáme normálu pro každou částici.

$$\mathbf{n}_i(\mathbf{r}) = \sum_{j=1}^{N} m_j \frac{1}{\rho_j} \nabla W(\mathbf{r}_i - \mathbf{r}_j, h)$$

Tato normála má nenulovou velikost pouze blízko povrchu. Pro křivost povrchu platí následující vztah

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

Pro výpočet κ pro každou částici budeme potřebovat laplacián obarvení, který vypočteme podle vztahu

$$\nabla^2 c_i(\mathbf{r}) = \sum_{j=1}^{N} m_j \frac{1}{\rho_j} \nabla^2 W(\mathbf{r}_i - \mathbf{r}_j, h).$$

Výsledný vztah pro povrchové napětí je

$$\mathbf{f}^{surface} = \sigma \kappa \mathbf{n},$$

tedy pro každou částici a po dosazení získáme vztah

$$\mathbf{f}_i^{surface} = -\sigma \nabla^2 c_i \frac{\mathbf{n}_i}{|\mathbf{n}_i|}.$$

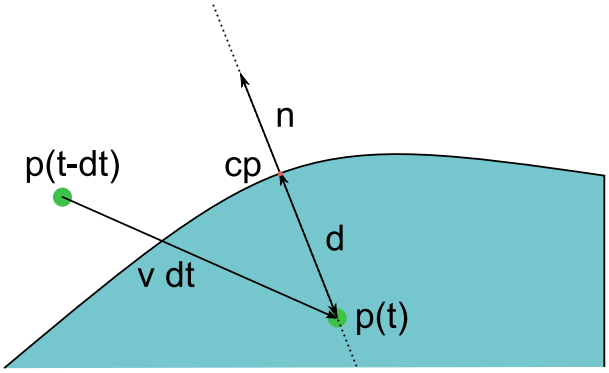
Vyhodnocení výrazu $\frac{\mathbf{n}_i}{|\mathbf{n}_i|}$ v místech, kde $|\mathbf{n}_i|$ je malé, vede k numerickým problémům. Také $|\mathbf{n}_i|$, které je malé, poukazuje na částici vzdálenou od povrchu, kde povrchové napětí nemá smysl. Proto $\mathbf{f}_i^{surface}$ vyhodnocujeme pouze pokud $|\mathbf{n}_i|$ přesáhne určitou mez, kterou označíme $t_{surface}$.

5.4 Kolize

Dalším aspektem simulace kapalin jsou kolize s okolím. Okolím může být stěna kontejneru obsahujícího celý objem kapaliny, překážka vložená do testovací scény či výšková mapa představující terén, ve kterém je chování kapaliny simulováno. Zajištění kolizí rozdělíme na dvě podúlohy - detekci kolize a reakce na kolizi.

5.4.1 Detekce kolize

Každá částice je nositelem pozice $\mathbf{r}$ a rychlosti $\mathbf{v}$, což jsou dostatečné údaje pro detekci jakékoli kolize. Pokud je potřeba zjistit, kde se nacházela částice před kolizí, stačí od aktuální pozice odečíst vektor rychlosti násobený časovým krokem Δt. Na obrázku 5.6 je diagram obecné kolize. Údaje které jsou potřeba pro správnou reakci na kolizi jsou:

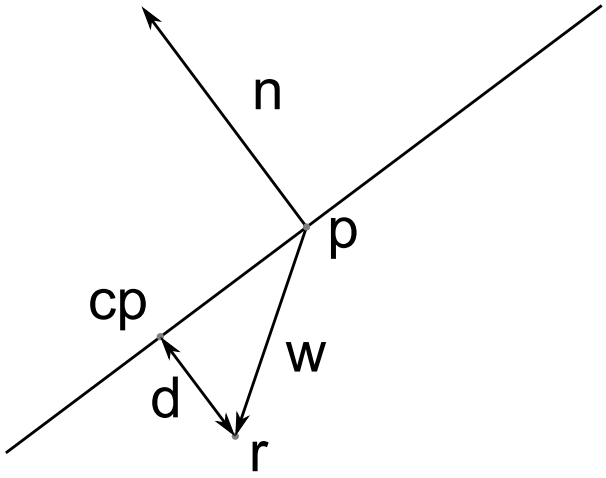


Obrázek 5.6: Diagram obecné kolize.

- Bod kontaktu s tělesem, **cp**.
- Hloubka průniku do tělesa, d
- Normála povrchu tělesa v místě bodu kontaktu, **n**.

Z obrázku je patrné, že používaný bod kontaktu nemusí být vždy bod, v němž spojnice předchozí a aktuální polohy částice protne povrch tělesa. Podle [4] můžeme použít nejbližší bod ve směru normály (což je výpočetně výhodné u implicitních těles), aniž to významně ovlivní výsledek kolize. V následujících podkapitolách jsou popsány způsoby získání potřebných údajů pro použitá tělesa.

5.4.1.1 Rovina



Obrázek 5.7: Diagram kolize s rovinou.

Na obrázku 5.7 je diagram kolize s rovinou. Rovina je definována bodem **p** a normálou **n**. Hloubkou průniku d, je vzdálenost částice od roviny. Tu vypočteme pomocí těchto vztahů

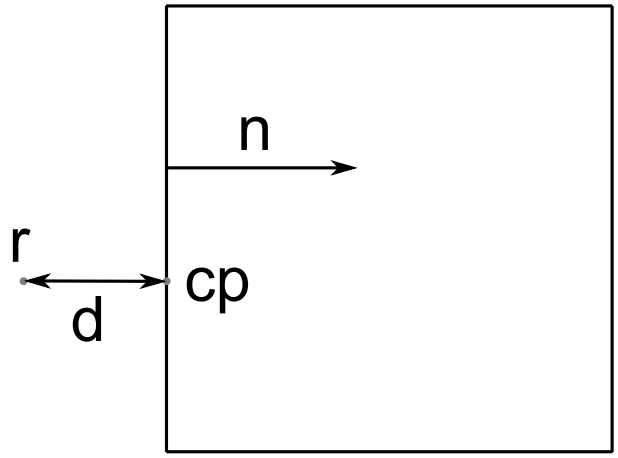
$$\mathbf{w} = \mathbf{r} - \mathbf{p},$$

$$d = -\mathbf{w} \cdot \mathbf{n},$$

a pro bod kontaktu platí vztah

$$\mathbf{cp} = \mathbf{r} + d\mathbf{n}.$$

5.4.1.2 Kvádr



Obrázek 5.8: Diagram kolize s kvádrem. 2D řez.

Na obrázku 5.8 je diagram kolize s kvádrem, při níž má být tekutina uvnitř kvádru. Kvádr je tedy *kontejnerem*. Kvádr je definován středem $\mathbf{p}$, vektorem velikostí $\mathbf{s}$ a maticí R popisující rotaci. Pro výpočet údajů popisujících kolizi nejprve převedeme pozici částice do lokální soustavy kvádru

$$\mathbf{r}_{local} = \mathsf{R}(\mathbf{r} - \mathbf{p}).$$

Poté získáme lokální souřadnici bodu kontaktu

$$\mathbf{cp}_{local} = \min(\mathbf{s}, \max(-\mathbf{s}, \mathbf{r}_{local})),$$

kde min a max jsou operace vyhodnocující každou složku vektoru zvlášť. Pro získání souřadnice bodu kontaktu v globálním souřadnicovém systému použijeme vztah

$$\mathbf{cp} = \mathbf{p} + \mathsf{R}\mathbf{cp}_{local},$$

a hloubku průniku pak snadno získáme

$$d = |\mathbf{cp} - \mathbf{r}|.$$

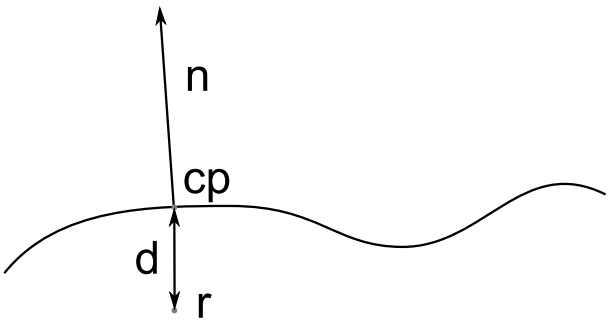
Poslední údaj, který potřebujeme pro zpracování kolize, je normála v místě dotyku. Tu získáme podle vztahu

$$\mathbf{n} = \mathsf{R}\mathrm{signum}(\mathbf{r}_{local} - \mathbf{cp}_{local}),$$

kde funkce signum pracuje po složkách vektoru.

5.4.1.3 Výšková mapa

Výšková mapa je 2,5D reprezentace terénu. Bývá implementována jako dvourozměrná matice. Každý údaj v matici je výška od vztažné roviny pro bod o daných souřadnicích $[x, y]$. Detailnější informace o implementaci výškové mapy jsou v kapitole 7.2.3. Na obrázku 5.9 je diagram kolize



Obrázek 5.9: Diagram kolize s výškovou mapou. 2D řez.

s výškovou mapou. Implementace výškové mapy je schopna pro libovolný bod $\mathbf{r}$, poskytnout výšku z a normálu povrchu $\mathbf{n}$. Výpočet údajů potřebných pro kolizi je pak přímočarý:

$$d = z(\mathbf{r}) - r_z$$

a

$$\mathbf{cp} = \mathbf{r} + [0, 0, d].$$

5.4.2 Reakce na kolizi

Po získání všech potřebných údajů o kolizi je potřeba modifikovat pozici a rychlost částice. Pozice částice se jednoduše přesune do bodu kontaktu, tedy

$$\mathbf{r}_i = \mathbf{cp}.$$

Pro inelastické srážky platí, že výsledná rychlost ve směru normály je nulová, zatímco pro plně elastické srážky platí, že výsledná rychlost ve směru normály je opačná. Kapaliny se chovají spíše inelasticky, ale je velmi jednoduché, zavést konstantu $0 \leq c_r \leq 1$, která bude určovat míru elasticity srážky. Rovnice pro rychlost částice by pak měla tvar

$$\mathbf{v}_i = \mathbf{v}_i - (1 + c_r)(\mathbf{v}_i \cdot \mathbf{n})\mathbf{n}.$$

Tato rovnice by v některých případech mohla, při nenulové konstantě c_r, zvýšit kinetickou energii částice. Proto je třeba zajistit, že budeme počítat pouze s energií, která odpovídá hloubce průniku. To zajistíme vložením poměru hloubky průniku ku vzdálenosti, kterou částice urazila od posledního kroku simulace. Výsledná rovnice má tvar

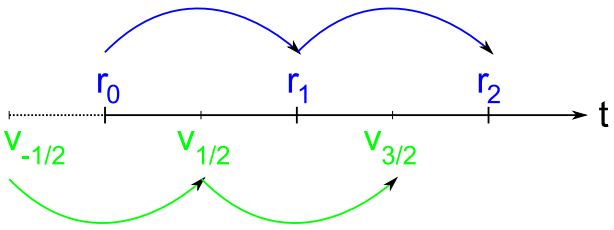
$$\mathbf{v}_i = \mathbf{v}_i - (1 + c_r \frac{d}{\Delta t |\mathbf{v}_i|})(\mathbf{v}_i \cdot \mathbf{n})\mathbf{n}.$$

5.5 Integrace

Při simulaci toku kapaliny je pozice a rychlost každé částice přepočítávána s časovým krokem Δt. Rovnice (5.2) je vyhodnocena a nová pozice částice je získána pomocí numerické integrace zrychlení. V této prácí je použit leap-frog integrátor, jehož činnost je znázorněna na obrázku 5.10. Pro tento integrátor platí následující vztahy

$$\mathbf{v}_{t+\frac{1}{2}\Delta t} = \mathbf{v}_{t-\frac{1}{2}\Delta t} + \Delta t \mathbf{a}_t,$$

$$\mathbf{r}_{t+\Delta t} = \mathbf{r}_t + \Delta t \mathbf{v}_{t+\frac{1}{2}\Delta t}.$$



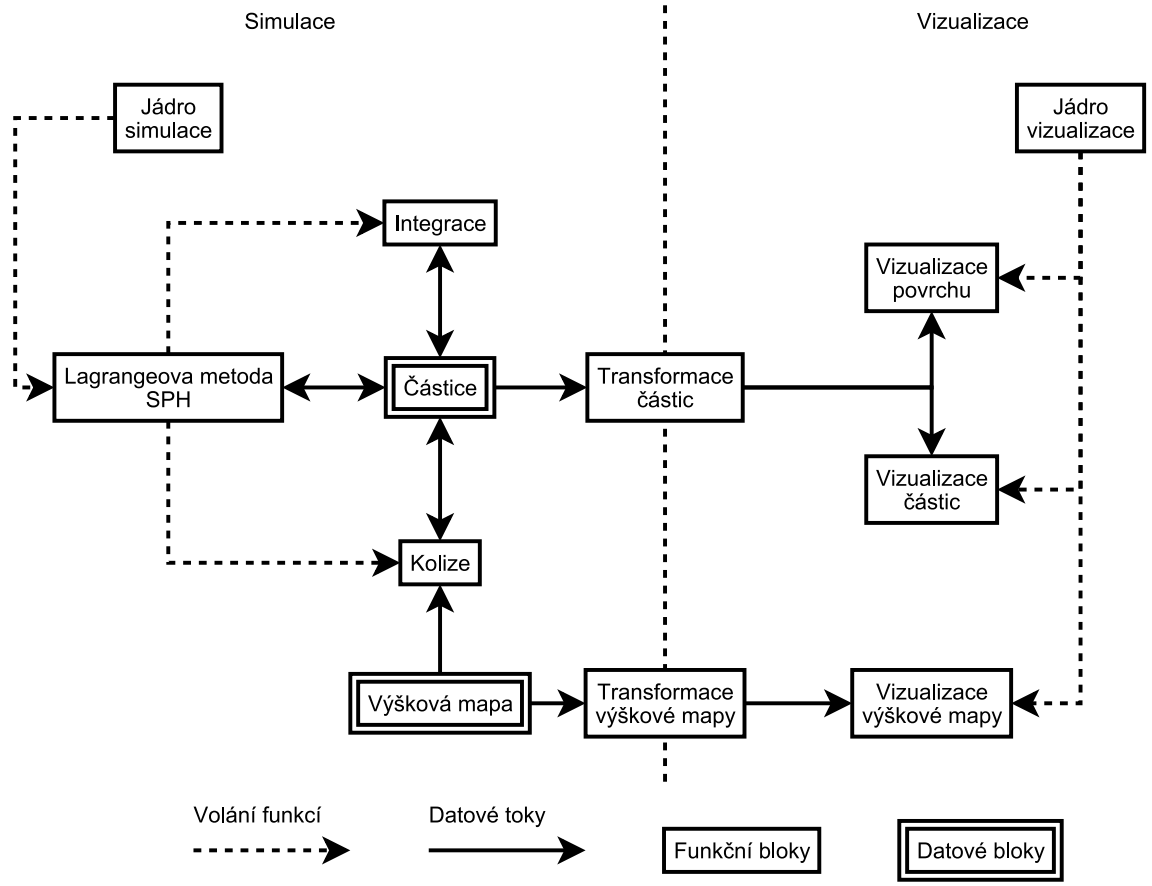
Obrázek 5.10: Činnost leap-frog integrátoru.

Rychlost v čase t kterou potřebujeme pro vyhodnocení silových polí, je získána pomocí následující aproximace

$$\mathbf{v}_t \approx \frac{\mathbf{v}_{t-\frac{1}{2}\Delta t} + \mathbf{v}_{t+\frac{1}{2}\Delta t}}{2}.$$

Podle [4] je tento integrátor nejvhodnější z hlediska stability a jednoduchosti implementace reakcí na kolize.

6 Návrh řešení



Obrázek 6.1: Návrh řešení.

V předchozích kapitolách jsme se seznámili s SPH a Lagrangeovou metodou pro modelování kapalin. Cílem této práce je také vizualizace stavových veličin a povrchu kapaliny. Úloha je tedy přirozeně rozdělena na část simulační a vizualizační. Na obrázku 6.1 je návrh řešení. V tomto diagramu jsou pohromadě funkční a datové bloky. Simulaci odpovídá levá strana diagramu, vizualizaci pravá, odděleny jsou přerušovanou čarou. Nejprve popíšeme datové bloky, které uchovávají data potřebná pro simulaci i vizualizaci.

- *Částice* jsou datový kontejner, obsahující vlastní částice. Každá částice obsahuje potřebné stavové veličiny. Tento kontejner také umožňuje vyhledávat částice v okolí dané částice.

- *Výšková mapa* obsahuje výšková data načtená ze souboru. Data jsou ve formě rovnoměrné mřížky a tento datový blok umožňuje dotazy na výšku a normálu povrchu v libovolném bodě.

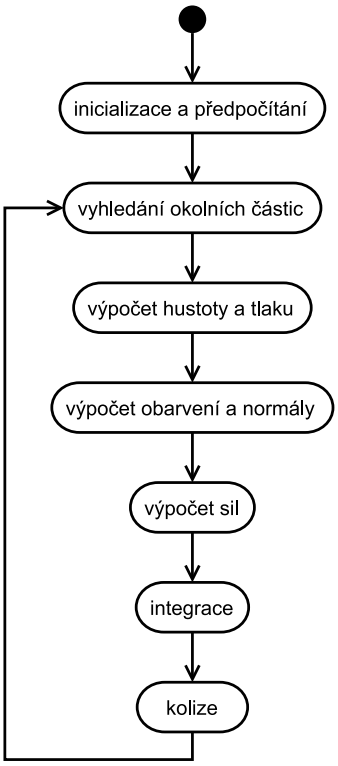
Funkční bloky obsahují funkcionalitu potřebnou pro vlastní simulaci a vizualizaci

- *Jádro simulace* je zodpovědné za řízení celé simulace, nastavování parametrů a komunikaci s okolím.

- *Lagrangeova metoda SPH* je vlastní implementace rovnic Lagrangeovy metody. Používá a modifikuje částice.

- *Integrace* je vlastní integrátor, aktualizuje pozici a rychlost částic podle sil vypočtených modelem.
- *Kolize* jsou spojením detekce a reakce na kolize, používají nejen částice, ale také data výškové mapy
- *Jádro vizualizace* podobně jako jádro simulace je zodpovědné za řízení vizualizace a komunikaci s uživatelem.
- *Vizualizace povrchu* zkonstruuje povrch kapaliny a zajistí jeho zobrazení.
- *Vizualizace částic* umožňuje zobrazení stavových veličin částic.
- *Transformace částic* převádí data částic do formy vhodné pro vizualizaci a slouží také jako rozhraní přes které spolu komunikuje vizualizační a simulační část.
- *Vizualizace výškové mapy* zkonstruuje povrch výškové mapy a zobrazí ho.
- *Transformace výškové mapy* je další rozhraní pro přenos dat mezi částmi. Zajistí také decimaci dat výškové mapy.

Ze vzájemných vztahů mezi stavovými veličinami je jednoznačně dán postup výpočtu. Diagram je na obrázku 6.2. Jelikož v krocích *výpočet hustoty a tlaku*, *výpočet obarvení a normály* a *výpočet sil* jsou ve výpočtech používána omezená jádra (viz kapitola 4.2), stačí vyhodnotit výrazy pouze pro částice do vzdálenosti h. Vyhledání těchto částic je provedeno v každém simulačním kroku pouze jednou, a to v kroku *vyhledání okolních částic*.



Obrázek 6.2: Diagram průběhu simulace.

7 Popis implementace

Pro implementaci jsem zvolil jazyk C++, pro vizualizaci je využito knihovny OpenGL [9]. Celý program je rozdělen do tří modulů, jednoho spustitelného souboru `Flow.exe` a dvou dynamických knihoven `Simulation.dll` a `Renderer.dll`. Jelikož dnes se čím dál častěji vyskytují osobní počítače s více než jedním procesorem, aplikace rozdělí simulační a vizualizační část každou do jednoho vlákna. Toto rozdělení klade zvýšené nároky na zajištění konzistence dat sdílených vizualizační a simulační částí. Toto se týká bloků *Transformace částic* a *Transformace výškové mapy* zmíněných v předchozí kapitole.
Zdrojový kód je rozdělen do šesti projektů:

- *common* obsahuje definice objektů sdílených v ostatních projektech, definice datových typů používaných pro předávání dat mezi moduly.
- *configs* obsahuje konfigurační soubory pro použité scény.
- *Flow* viz dále.
- *img* obsahuje použité textury a výškové mapy.
- *Renderer* viz dále.
- *Simulation* viz dále.

V následujících kapitolách je popis tří projektů tvořících moduly výsledného programu.

7.1 Flow

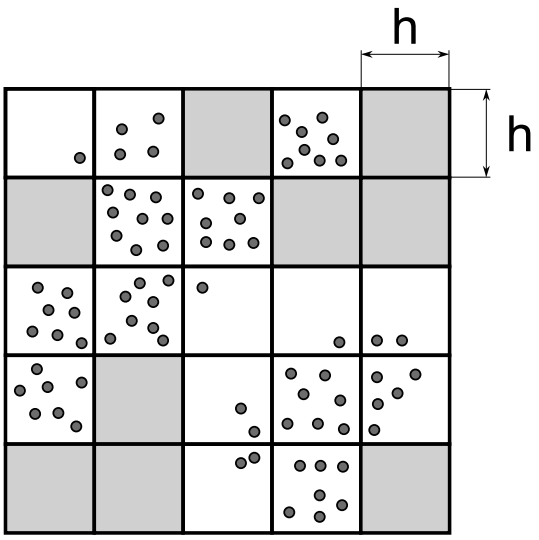
Flow je projekt, jehož překladem vznikne spustitelný soubor `Flow.exe`. V tomto modulu se vytvoří okno pro výstup vizualizace, simulační vlákno, načtou se proměnné z konfiguračního souboru a zinicializují se data sdílená moduly `Simulation.dll` a `Renderer.dll`.

7.2 Simulation

Simulation je projekt, jehož překladem vznikne dynamická knihovna `Simulation.dll`. V tomto modulu je vlastní kód simulace, tedy vše, co je na levé straně diagramu na obrázku 6.1, a také kód odpovídající blokům *Transformace částic* a *Transformace výškové mapy*, které zajišťují předávání simulačních dat vizualizaci. Hlavní třídou je `CSimulationCore`, která obsahuje funkčnost bloků *Jádro simulace*, *Lagrangeova metoda SPH*, *Integrace* a *Kolize*. Metoda `Step` této třídy provede jeden časový krok simulace. Instance této třídy je uložena v globální proměnné `g_simulation`. Třída `CKernels` je těsně spjata s touto třídou, je implementací používaných jader, jejich gradientů a laplaciánů. Metoda `Precompute` slouží k předpočítání koeficientů závislých na vyhlazovací délce h. V následujících podkapitolách jsou probrány nejdůležitější části implementace modulu `Simulation.dll`.

7.2.1 Hledání sousedních částic

Pro vyhodnocení sil a stavových proměnných je potřebné znát okolní částice daného místa. Toto je kritickým místem simulací založených na Lagrangeovské metodě, jelikož je obecně potřeba vyhodnotit vzájemné působení všech částic, což určuje složitost $O(n^2)$, kde n je počet částic. Při použití SPH a omezených jader jsou pak příspěvky od částic vzdálenějších než je vyhlazovací délka h nulové. Je tedy potřeba efektivně vyhledat částice bližší než h. Proto prostor rozdělíme do pomyslné 3D mřížky, jejíž buňky jsou krychle o hraně h. Částice ležící blíže než h poté



Obrázek 7.1: Buňky pomocné mřížky pro urychlení vyhledávání sousedních částic. Šedě vyplněné buňky nejsou uloženy v datové struktuře, protože neobsahují žádné částice.

musí ležet v buňce odpovídající danému místu nebo v jedné z 26 okolních buněk. Celočíselné souřadnice buněk této mřížky, implementuje třída `CBucketCoordinate`. Metoda `Set` této třídy převádí souřadnice světové na celočíselné souřadnice mřížky podle vztahu

$$x_{cell} = \begin{cases} \lfloor x \rfloor - 1 & x < 0 \\ \lfloor x \rfloor & jinak \end{cases},$$

analogicky pro souřadnice y a z. Buňka mřížky je tedy identifikována celočíselnou souřadnicí a obsahuje seznam částic jejichž pozice má stejné celočíselné souřadnice. Jelikož předem neznáme prostor, v němž se mohou částice pohybovat, není možné vytvořit všechny buňky. Použil jsem proto asociativní pole `std::map`, kde klíčem je celočíselná souřadnice a vlastními daty je lineární seznam částic náležející do dané buňky. Zjednodušené schéma je na obrázku 7.1. Pouze buňky obsahující částice budou v asociativním poli uloženy. Tuto datovou strukturu implementuje třída `CBucketMap`. Metody `Insert` a `Remove` vkládají a odstraňují částici a metoda `GetNeighborhood` vrátí seznam sousedních buněk pro zadanou celočíselnou souřadnici.
Při implementaci této datové struktury jsem narazil na pomalost standardního alokátoru `std::allocator<T>` používaného datovými strukturami z knihovny `STL`[1]. Alokátor je objekt, který alokuje a uvolňuje paměť pro prvky přidávané a odebírané do těchto struktur. Standardní alokátor používá funkce `malloc` a `free`, které jsou zbytečně obecné. Proto jsem implementoval vlastní alokátor, který přiděluje paměť z předem alokovaného bloku, pracuje s objekty o stejné velikosti a provádí alokaci i uvolnění v konstantním čase. Tento alokátor je implementován šablonami `CSimpleAllocator<T>` a `CSimpleStorage<T>`.

7.2.2 Data částice

Každá částice je reprezentována instancí objektu `CParticle`. Tento objekt má řadu datových atributů, z nichž některé jsou statické, společné pro všechny částice.

- `m_mass` hmotnost částice, odpovídá m.
- `m_rest_density` klidová hustota částice, odpovídá ρ_0.

[1] Standard Template Library [6]

- `m_stiff` konstanta tuhosti kapaliny, odpovídá k.
- `m_viscosity` viskozita kapaliny, odpovídá μ.
- `m_radius` poloměr částice. Je to takový poloměr, jaký by měla koule hmotnosti m a hustoty ρ_0.
- `m_surface_tension_coef` koeficient povrchového napětí, odpovídá σ.
- `m_surface_tension_threshold` mez povrchového napětí. Pokud má normála menší velikost než je tato mez, síla daná povrchovým napětím se nevyhodnocuje. Odpovídá $t_{surface}$.
- `m_pos` pozice částice, uchovávána jako 3D vektor implementovaný třídou `C3DVector`. Odpovídá $\mathbf{r}$.
- `m_velocity` rychlost částice v čase stejném jako je udána pozice. Odpovídá $\mathbf{v}$.
- `m_velocity_prev` rychlost částice v čase $t - \frac{1}{2}\Delta t$. Odpovídá $\mathbf{v}_{t-\frac{1}{2}\Delta t}$.
- `m_density` hustota v místě částice. Odpovídá ρ.
- `m_pressure` tlak v místě částice. Odpovídá p.
- `m_c` obarvení částice. Odpovídá c.
- `m_normal` vnitřní normála povrchu v místě částice. Odpovídá $\mathbf{n}$.
- `m_c_laplace` laplacián obarvení částice. Odpovídá $\nabla^2 c$.
- `m_force` výsledná síla působící na částici, bez započtení gravitace. Odpovídá $\mathbf{F}$.
- `m_force_visc` výsledná síla působící na částici způsobená viskozitou. Odpovídá $\mathbf{f}^{visc}$.
- `m_force_press` tlaková síla působící na částici. Odpovídá $\mathbf{f}^{press}$.
- `m_force_surf` výsledná síla působící na částici způsobená povrchovým napětím. Odpovídá $\mathbf{f}^{surface}$.
- `m_ID` identifikační číslo částice.
- `m_bucket_pos` iterátor do seznamu částic v buňce mřížky. Slouží pro rychlé odebrání částice z buňky v konstantním čase.
- `m_bucket_coord` celočíselná souřadnice buňky do které částice náleží.
- `m_neighbours` seznam sousedů částice bližších než h.
- `m_neighbours_dist` seznam vzdáleností k sousedům částice bližších než h.

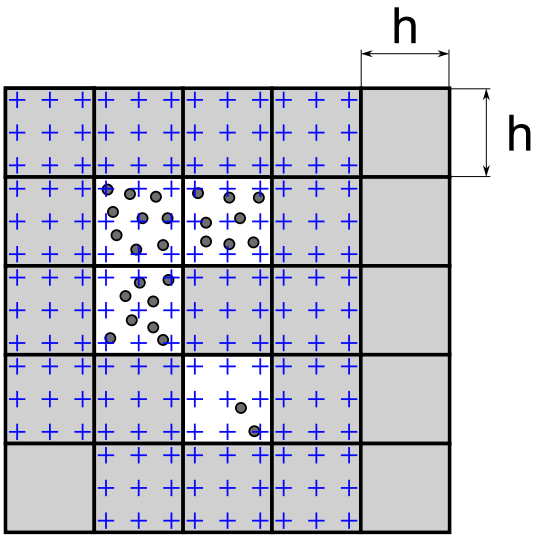
Datový blok *Částice* z předchozí kapitoly je tvořen dynamickým polem instancí této třídy. Toto pole je pak datovým atributem třídy `CSimulationCore`.

7.2.3 Výšková mapa

Poslední důležitou třídou je `CHeightMap`, která implementuje datový blok *Výšková mapa*. Objekt této třídy umí načíst obdélníkový obrázek ve stupních šedi představující výškovou mapu. Podporovaným datovým formátem je nekomprimovaný `TGA` soubor. Tato data jsou poté transformována do souřadnic používaných v simulaci, kde souřadnice z určuje výšku. Při načítání jsou také spočítány normálové vektory povrchu výškové mapy. Metody `GetAt` a `GetNormalAt` slouží pro získaní výšky respektive normály v bodě určeném souřadnicemi x a y. Hodnoty mimo body načtené ze souboru jsou dopočítány bilineární interpolací. Používaná výšková mapa je uchována v globální proměné `g_heightmap`.

7.2.4 Rozhraní mezi simulací a vizualizací

V předchozí kapitole, byly popsány dva funkční bloky *Transformace částic* a *Transformace výškové mapy* představující rozhraní pro přenos dat ze simulační části do části vizualizační. Tyto bloky jsou implementovány globálními funkcemi, které modul `Simulation.dll` exportuje. Tyto funkce synchronizují přístup k datům používaným simulační částí pomocí *mutexů* implementovaných třídou `CLockableBase`, která využívá mutexy Win32 API. Třídy, které odpovídají datovým blokům z předchozí kapitoly `CHeightMap` a `CSimulationCore`, jsou poděděny z třídy `CLockableBase` a umožňují tedy bezpečné sdílení dat mezi více vlákny, pokud je před každou operací volána metoda `Lock` a po dokončení metoda `UnLock`.
Blok *Transformace výškové mapy* je implementován funkcí `GetHeightMapPoints`, která vrátí žádaný počet bodů popisující výškovou mapu. Tyto body jsou rozmístěny po výškové mapě nerovnoměrně. Větší hustota je v místech, kde dochází k velké změně normály. Tato funkčnost byla implementována pro možnost nastavení kvality vizualizace výškové mapy. Kvalitnější vizualizace má vyšší nároky na rychlost zobrazovacího zařízení.

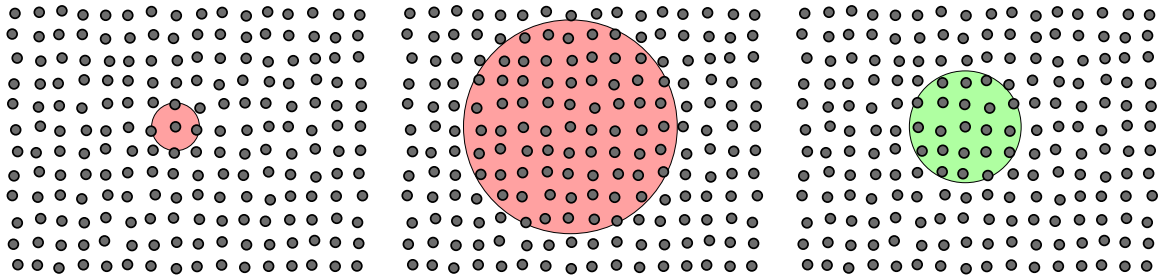


Obrázek 7.2: Vzorkování funkcí `GetPointInfo`.

Blok *Transformace částic* je implementován dvěma funkcemi. Funkce `GetParticles` vrací seznam instancí třídy `CGLParticleBase`, který obsahuje informace důležité pro vizualizaci částic (pozici, rychlost, tlak, hustotu, síly atd.). Druhá funkce `GetPointInfo` je určena pro vizualizaci povrchu kapaliny. Tato funkce vrátí seznam vzorků o žádaném rozlišení, pokrývající všechny buňky mřížky obsahující alespoň jednu částici a všechny buňky s těmito buňkami sousedící. Každý vzorek obsahuje hustotu a normálu povrchu v daném místě. Na obrázku 7.2 je

2D příklad vzorkování. Modrými křížky jsou znázorněny vzorkovací body, šedá kolečka jsou částice, bíle čtverce jsou buňky mřížky obsahující alespoň jednu částici.

7.2.5 Určení vyhlazovací délky



Obrázek 7.3: Volba velikosti vyhlazovací délky jádra. Na levém obrázku je h příliš malé, na prostředním příliš velké a na pravém je h zvoleno správně.

Vyhlazovací délka h je klíčovým parametrem jádra, který ma vliv na stabilitu i rychlost simulace. Na obrázku 7.3 jsou tři situace pro různé vyhlazovací délky. Na levém obrázku je h příliš malé - v okolí pak bude příliš málo částic a model přestane stabilně fungovat. Na prostředním obrázku je h příliš velké - v okolí je příliš mnoho částic, což má za následek nejen zpomalení simulace, ale také problémy se stabilitou v situacích, když je částice blízko povrchu kapaliny (například při vygenerování počátečního stavu). V takových situacích je okolí částice z velké části prázdné, což se projeví velkým poklesem hustoty. Na pravém obrázku je vyhlazovací délka přiměřená, okolí obsahuje rozumný počet částic, které zajistí funkčnost modelu při zachování rychlosti. Jelikož velikost h závisí na dalších parametrech kapaliny, program si hodnotu h vypočte sám z ostatních parametrů podle vztahu

$$h = \sqrt[3]{25\frac{3m}{4\pi\rho_0}},$$

kde 25 je konstanta zvolená podle [4] a odpovídá počtu částic v okolí velikosti h pro kapalinu v klidu s hustotou rovnou klidové hustotě ρ_0. Tento způsob je velmi jednoduchý a nezávislý na stavu kapaliny v konkrétním místě. [14] zmiňuje volbu h v závislosti na aktuální hustotě v místě částice.

7.2.6 Volba tuhosti a viskozity kapaliny

Všechny konstanty, které určují vlastní velikost sil působící na částice, potencionálně ovlivňují stabilitu řešení. Podle Courant-Friedrich-Lewyho podmínky [7] musí být časový krok tak malý, aby se změny v kapalině stačily propagovat do okolí. Při vysoké rychlosti částic, která je zapříčiněna velkými silami, je nutné zamezit časovému kroku tak velkému, že by se během něj částice mohly následkem vzájemné interakce pohnout o více než vyhlazovací délku h. Pro interaktivní simulaci je zapotřebí časový krok v řádu milisekund, proto je potřeba volit konstanty tak, aby simulace jako celek byla při tomto časovém kroku stabilní. Konstanta tuhosti k určuje míru stlačitelnosti kapaliny. Pro simulaci reálně stlačitelné kapaliny by tato konstanta byla příliš veliká a výsledný časový krok příliš malý. Proto jsou simulované kapaliny stlačitelné, viz kapitola 8.1.
Konstanta viskozity μ působí v systému jako tlumení, které snižuje energii systému. Proto by se mohlo zdát, že při vysoké viskozitě může být časový krok a tuhost kapaliny vysoká, ale není tomu tak. Pokud příspěvky od síly $\mathbf{f}^{visc}$ převáží nad ostatními silami, energie systému se

bude naopak zvyšovat, což bude mít za následek nestabilitu. Proto musí být konstanty k a μ v rovnováze. Analytické vyjádření rozmezí hodnot by bylo složité, proto nezbývá než zkoušet různé konstanty, což je ovšem u fyzikálních simulací běžné.
Při tomto zkoušení jsem také odhalil, že konstanta k závisí na hmotnosti částice. Čím je částice hmotnější, tím vyšší může být tuhost. Pro toto chování jsem nenalezl uspokojivé vysvětlení. Jedna z možných příčin je použití různých jader pro tlakovou sílu a pro viskozitu, kdy každé jádro má jiný průběh a při změně hmotnosti částice dochází také ke změně vyhlazovací délky h a pro stejně vzdálené body se příspěvky dané těmito jádry změní v jiném poměru.

7.3 Renderer

Renderer je projekt, jehož překladem vznikne dynamická knihovna `Renderer.dll`. V tomto modulu je kód vizualizace využívající OpenGL a také kód okna aplikace. Hlavní třídy jsou `CGLRenderer` řídící vykreslování, `CGLWindow` vytvářející okno aplikace a obstarávající interakci s uživatelem a `CGLConsole` umožňující vyvolání příkazů a změny některých parametrů za běhu aplikace. Instance těchto tříd jsou uloženy v globálních proměných `g_renderer`, `g_MainWnd` a `g_console`. Jaké příkazy a jaké parametry, to vysvětlí následující podkapitola.

7.3.1 Příkazy a parametry

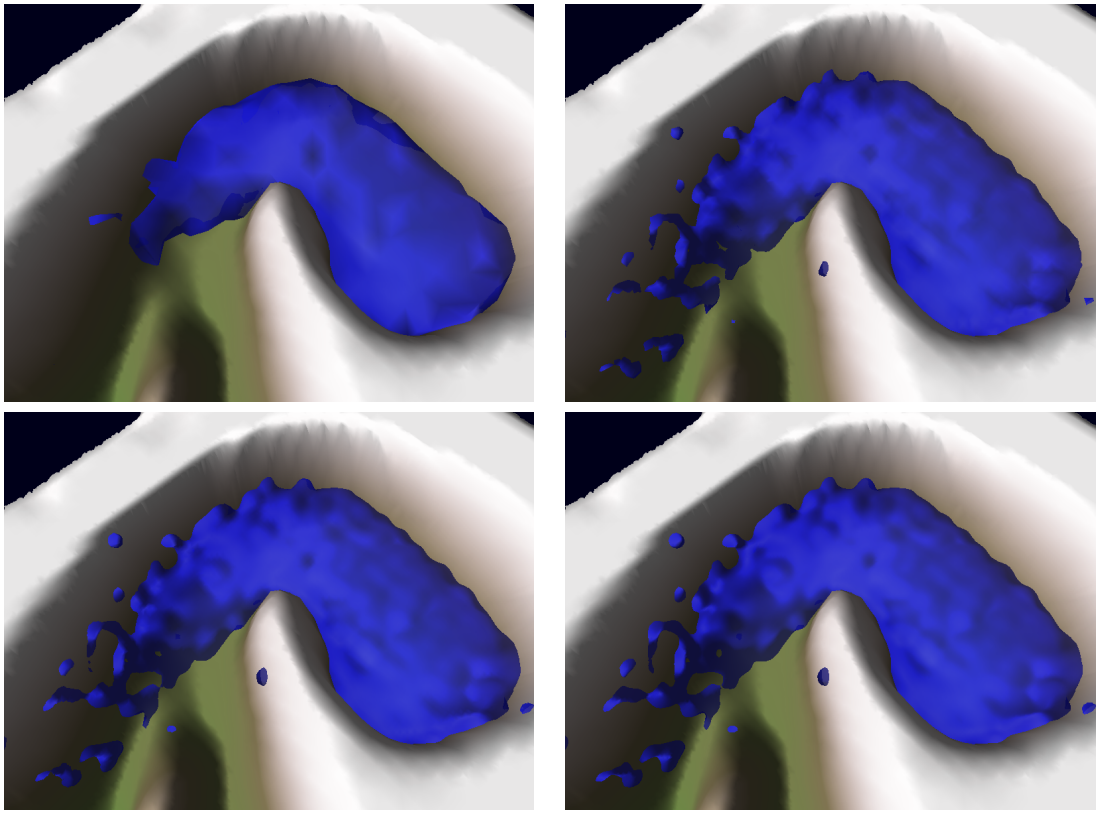
Při vlastní simulaci je potřeba množství konstant (jako hmotnost částice, časový krok atd.). Tyto konstanty je možné zapsat přímo do zdrojového kódu, což je řešení jednoduché (stačí zavést jeden hlavičkový soubor s definicí těchto konstant), ale nepraktické. Při změně konstanty je třeba znovu přeložit celý program. Lepším řešením by bylo zapsat konstanty v textovém tvaru do konfiguračního souboru a ten při spuštění aplikace načíst. Zde je pro změnu konstanty třeba pouze upravit konfigurační soubor a znovu spustit aplikaci. Použité řešení jde ještě o krok dále a přidává možnost měnit konstanty za běhu programu. Základem je třída `CConsoleItemBase`, která odpovídá jednomu parametru nebo příkazu. Instance této třídy je jednoznačně identifikována svým jménem. Metoda `Perform` slouží pro provedení akce (přiřazení hodnoty, provedení příkazu). Poděděné třídy přidají vlastní datový typ (booleovskou hodnotu, celé číslo, řetězec atd.) a implementují metodu `Perform`. Každý příkaz odpovídá jedné třídě a její instance je staticky vytvořena již při kompilaci. Parametr může být též vytvořen staticky (například parametr *sim_t* představující simulační čas), nebo jako většina vytvořen dynamicky za běhu aplikace podle obsahu konfiguračního souboru. Struktura tohoto souboru je popsána v dodatku B, seznam parametrů a příkazů je v dodatku C. Všechny instance, vytvořené staticky nebo dynamicky, jsou uloženy v kontejneru implementovaném třídou `CVariableContainer`. Tento kontejner je datovým atributem třídy `CGLConsole`, která implementuje několik metod pro pohodlný přístup k parametrům. Jelikož je ukazatel na globální proměnou obsahující instanci třídy `CGLConsole` exportovaný z modulu, parametry jsou přístupné z celé aplikace.

7.3.2 Vizualizace povrchu

Pro vizualizaci povrchu jsem zvolil algoritmus *pochodující krychle (marching cubes)*. Tento algoritmus používá vzorky v pravidelné trojrozměrné mřížce. Toto vzorkování je popsáno v kapitole 7.2.4. Pro získání triangulace povrchu je třeba určit hraniční hodnotu, která rozdělí vzorky na vnitřní a vnější. Tato hodnota je parametrem *r_mc_iso*. Jemnost mřížky je určena parametrem *r_mc_subdiv* který určí délku hrany mřížky podle vztahu

$$d_{mc} = \frac{h}{r_mc_subdiv}.$$

Jemnost mřížky má zásadní vliv na kvalitu výsledného povrchu. Na obrázku 7.4 je patrný vliv jemnosti mřížky na kvalitu nalezeného povrchu. Jelikož algoritmus používá lineární interpolaci,



Obrázek 7.4: Vliv parametru *r_mc_subdiv* na kvalitu výsledného povrchu. Postupně pro hodnoty 1, 3, 5 a 8.

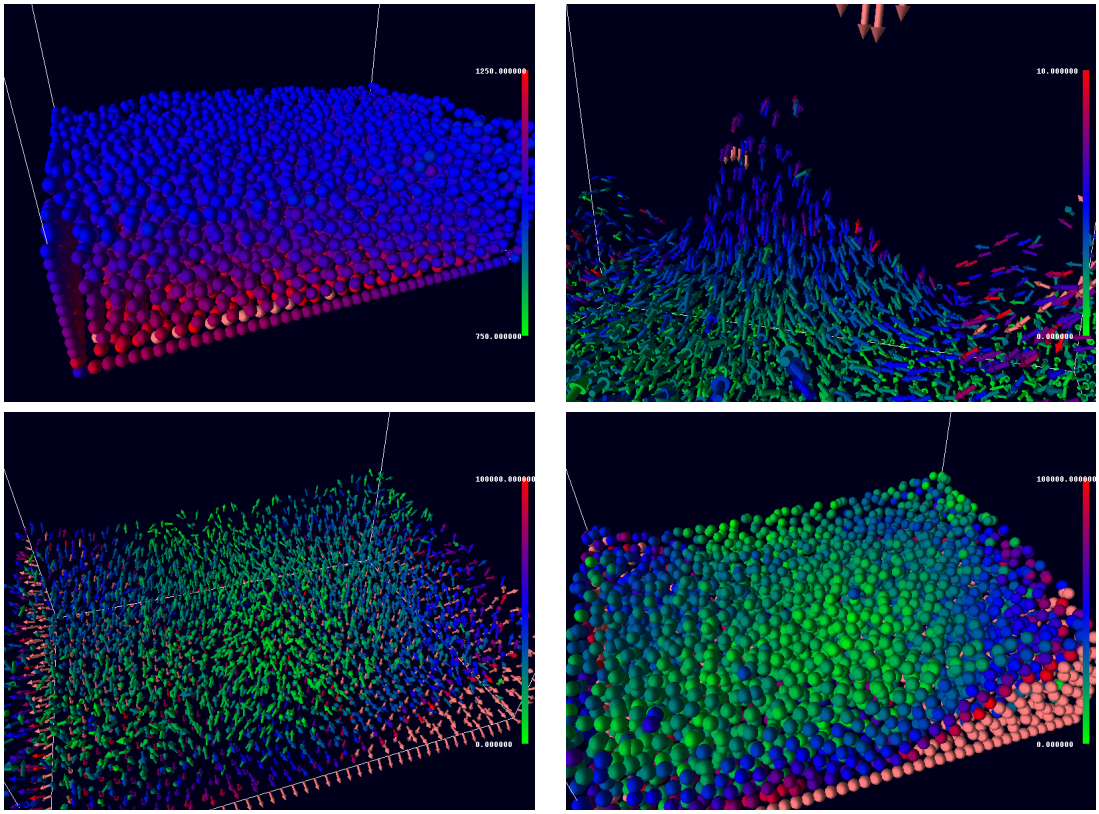
je rozdíl mezi hodnotou parametru *r_mc_subdiv* 1 a 3 velmi zřetelný. Čas potřebný pro vizualizaci povrchu je rozdělen mezi získání vzorků a vlastní běh algoritmu pochodující krychle. Výsledky jsou v tabulce 7.1. Je zřejmé, že pro jemnější mřížku je vzorkování neúnosně časově náročné. Počet vzorků roste s třetí mocninou parametru *r_mc_subdiv*. Jedná se v podstatě o přechod z reprezentace založené na částicích na reprezentaci založenou na mřížce. Proto je algoritmus pochodující krychle vhodnější pro Eulerovu metodu. [7] s úspěchem použil algoritmus *naplácávání (point splatting)*, který vychází přímo z dat uchovaných v každé částici a je tudíž řádově rychlejší.

7.3.3 Vizualizace stavové veličiny

Vizualizace stavových veličin je jedním z hlavních cílů této aplikace. Data, která se vizualizují, jsou stavové veličiny, které jsou vypočteny pro potřebu modelu kapaliny pro každou částici. Tato

r_mc_subdiv	vzorkování [ms]	pochodující krychle [ms]
1	22	<1
3	110	16
5	342	31
8	1117	78

Tabulka 7.1: Časy potřebné pro vzorkování a běh algoritmu pochodující krychle pro různou jemnost mřížky.



Obrázek 7.5: Různé druhy vizualizací.

data jsou získána pomocí funkce `GetParticles` exportované modulem `Simulation.dll`. Data jsou uchována v instanci třídy `CGLParticleBase`. Veličiny, které chceme vizualizovat, se dají rozdělit na vektorové a skalární. Skalární veličiny jsou vizualizovány koulemi na pozicích částic. Koule jsou obarveny podle hodnoty zobrazované veličiny. Aplikace umožňuje použít libovolnou stupnici barev definovanou obrázkem. Je možno použít libovolný obrázek uložený v souboru `img/gauge.tga` s výškou nebo šířkou 1 pixel. Obrázek je použit jako 1D textura a hodnota texturovací souřadnice je vypočtena podle vztahu

$$s = \frac{v - v_{min}}{v_{max} - v_{min}},$$

kde s je hodnota texturovací souřadnice, v hodnota veličiny, kterou chceme zobrazit, v_{min} a v_{max} jsou hodnoty parametrů nastavitelné v konfiguračním souboru. Na obrázku 7.5 jsou příklady vizualizace skalárních veličin, na levém horním obrázku je vizualizována hustota, na pravém spodním velikost tlakové síly $\mathbf{f}^{\mathbf{visc}}$.
Vektorové veličiny, jsou znázorněny 3D šipkou, která směřuje ve směru vektoru. Pokud je velikost vektoru menší než 10^{-5}, jsou znázorněny malou koulí. Barva šipky, podobně jako u skalární veličiny, odpovídá velikosti vektoru podle vztahu

$$s = \frac{|\mathbf{v}|}{v_{scale}},$$

kde $\mathbf{v}$ je vektor který vizualizujeme a v_{scale} je nastavitelný parametr. Na obrázku 7.5 jsou vizualizována vektorová data. Na obrázku vpravo nahoře jde o rychlosti a na obrázku vlevo dole o tlakovou sílu. Za pozornost stojí porovnání dolních obrázků, které zobrazují vektorovou veličinu a její velikost.

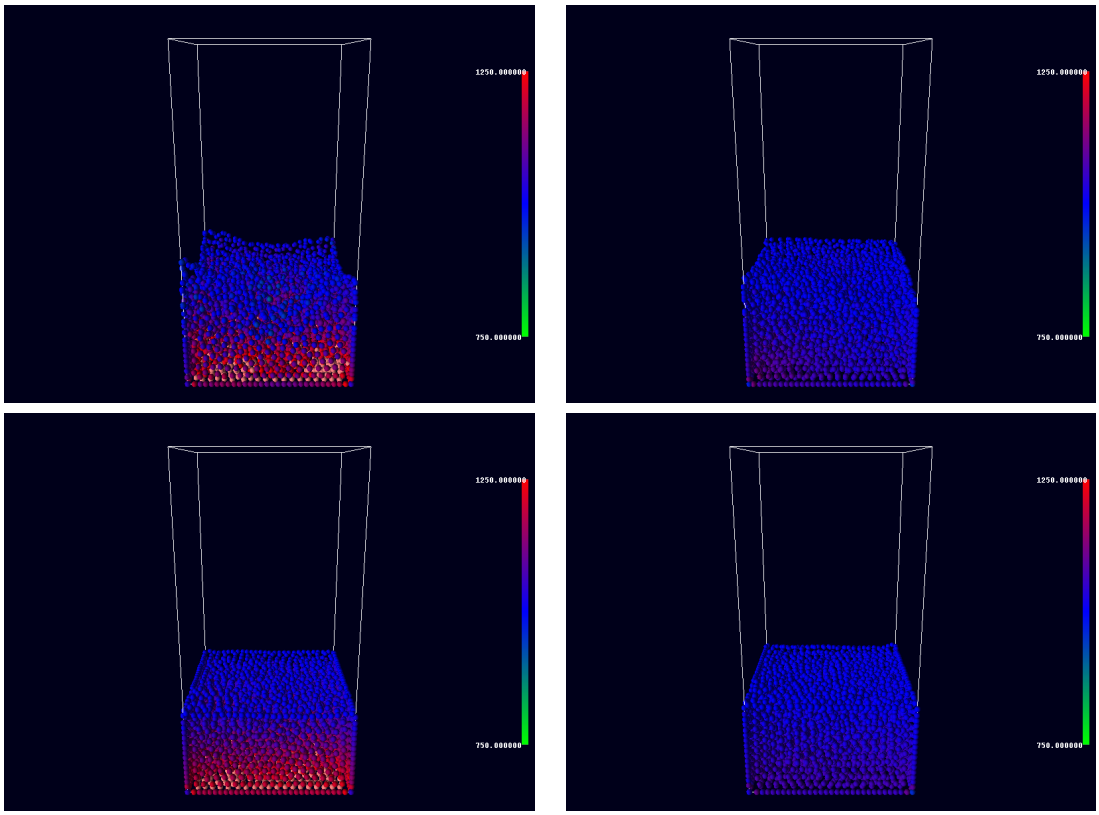
7.3.4 Vizualizace výškové mapy

Vizualizace výškové mapy je založena na *Delaunayově triangulaci* množiny bodů vrácených funkcí `GetHeightMapPoints`, která je popsána v kapitole 7.2.4. Je použita inkrementální metoda, která má časovou složitost $O(n^2)$. Triangulace proběhne typicky pouze jednou a její výsledek je uložen. Při vykreslování se použije 1D textura uložená v souboru `img/hory.tga` a texturovací souřadnice je navázána na výšku. Výsledek je vidět na obrázku 7.4.

8 Testy a výsledky

V této kapitole je ukázán vliv nejdůležitějších parametrů na výsledné chování a některé pozoruhodné aspekty chování simulované kapaliny. V závěru je představena časová a paměťová složitost.

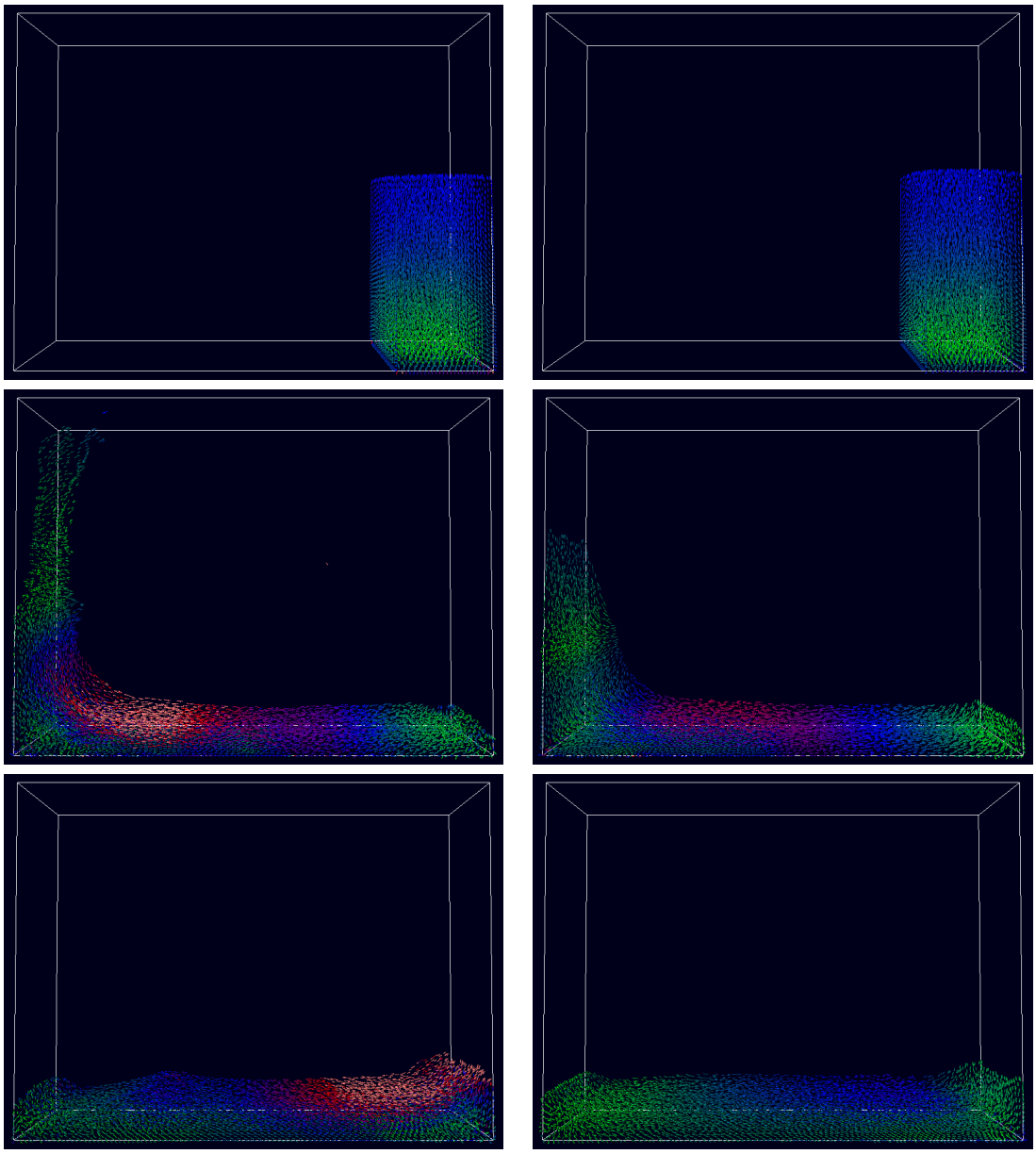
8.1 Konstanta tuhosti k



Obrázek 8.1: Testovací scéna pro znázornění vlivu konstanty tuhosti. Kapalina na levé straně má konstantu tuhosti $k = 1000$, na pravé $k = 3000$. Snímky odpovídají stavu po 5s.

Jak již bylo zmíněno v kapitole 7.2.6, konstanta tuhosti k určuje míru stlačitelnosti kapaliny. Na obrázku 8.1 jsou snímky stejné scény pro různé konstanty k. Snímky vlevo odpovídají $k = 1000$, vpravo $k = 3000$. Ostatní parametry jsou shodné. Na snímcích je vizualizována hustota. Modrá barva odpovídá hustotě $1000\frac{kg}{m^3}$, což je hodnota konstanty ρ_0 - tedy žádaná hustota. Čím je barva červenější, tím je hustota větší. Na snímcích je zřejmé, že pro nižší tuhost dochází v kapalině k větším odchylkám hustoty od hodnoty ρ_0 a kapalina se tedy jeví stlačitelnější. Pro simulaci kapalin, které se blíží fyzikálními vlastnostmi realitě, je třeba volit konstantu k velmi vysokou, což by mělo za následek malý časový krok a tedy pomalou simulaci. Při vlastních testech byl časový krok simulace $10ms$ a konstanta tuhosti byla zvolena tak velká, aby simulace zůstala stabilní a došlo k ustálení.
Dalším jevem, na který má tuhost vliv, je amplituda a frekvence kmitání kapaliny při ustalování. Toto je velmi dobře patrné na videosekvencích `box_stiff_1000.avi` a `box_stiff_3000.avi`, které jsou součástí přílohy této práce. Čím je tuhost vyšší, tím je amplituda kmitů menší a frekvence vyšší. Toto je analogické s harmonickým pohybem pružiny, která má podobný vztah, jako tlaková síla.

8.2 Konstanta viskozity μ

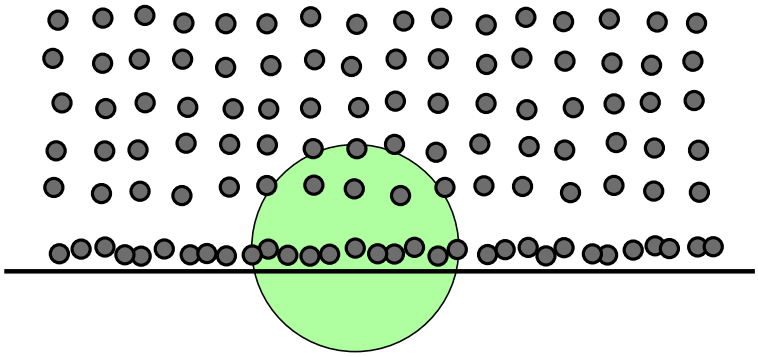


Obrázek 8.2: Testovací scéna pro znázornění vlivu viskozity. Kapalina na levé straně má vizkozitou $\mu = 1000$, na pravé $\mu = 5000$. Snímky odpovídají stavu po 3s.

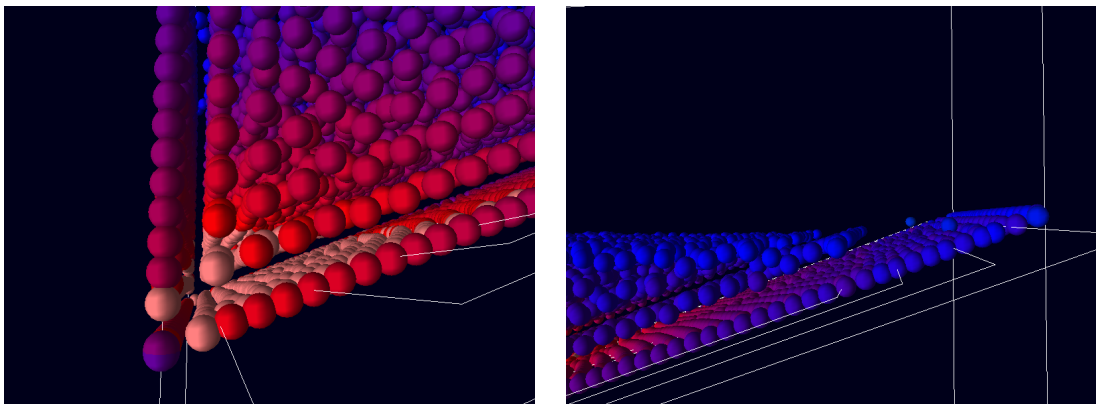
V kapitole 7.2.6 je popsán význam viskozity pro stabilitu simulace a její vztah ke konstantě tuhosti. Viskozita také velkou měrou ovlivňuje vlastní chování kapaliny. Na obrázku 8.2 jsou snímky scény, kdy byla kapalina uzavřena v pravé části a po ustálení byla uvolněna. Na snímcích je vizualizace rychlostí částic. Zelená barva odpovídá nulové rychlosti, modrá rychlosti $5\frac{m}{s}$ a červená $10\frac{m}{s}$. Snímky jsou pro časy 0, 3 a $6s$ po uvolnění. Levé snímky jsou pro kapalinu s viskozitou $\mu = 1000$, pravé $\mu = 5000$. Výchozí situace je pro oba případy srovnatelná. Již v druhém řádku, který odpovídá situaci v čase $3s$, je patrný celkově pomalejší pohyb s menší maximální rychlostí. Kapalina s větší viskozitou má také celkově menší energii, což je patrné z výšky sloupce u levé stěny. Ve třetím řádku se šíří zpětná vlna zleva. Rozdíl rychlosti šíření této vlny mezi kapalinami s různou viskozitou je již velmi zřetelný.

Ve videosekvencích `box_visc_1000.avi` a `box_visc_5000.avi` je také patrný rozdíl v soudržnosti kapaliny při působení externích sil zastoupených kolizemi se stěnami kontejneru.

8.3 Chování na hranicích



Obrázek 8.3: Diagram chování kapaliny na hranici. Šedé kruhy představují částice, zelený okolí jedné částice.



Obrázek 8.4: Testovací scéna pro znázornění chování na hranicích.

Implementovaná metoda na hranicích kolizních objektů vede ke zvýšení hustoty částic. Na obrázku 8.3 je zjednodušené schéma situace blízko hranice představované na obrázku horizontální úsečkou. Částice poblíž hranice mají část svého okolí o velikosti h na druhé straně hranice, kde nemohou být žádné částice. Aby kapalina dosáhla žádané hustoty ρ_0, musí být v okolí dostatek částic, proto na hranici dojde k zahuštění částic. Toto zahuštění je zvlášť patrné u hranic definovaných rovinou. Částice, které nejsou součástí vrstvy na hranici, pak musejí být vzdálenější než uvnitř objemu, protože jinak by ve svém okolí měly příliš částic ze zahuštěné vrstvy na hranici, což by vedlo k vysoké hustotě a výsledná tlaková síla by způsobila oddálení. Tento jev se také objevuje ve vrstvě tvořící volnou hladinu. V kontejnerech které jsou složené z rovin, dochází k "vysunutí" vrstvy na hranici nad volnou hladinu. Toto je vidět na obrázku 8.4. Jedna z cest k omezení tohoto jevu je generování částic na druhé straně hranice, které by sloužily pro výpočet hustoty, případně korekce hustoty částic na hranici.

n	m	h	k
1000	800	1.683890	3000
2000	400	1.336505	3000
4000	200	1.060784	1000
6000	133	0.925908	2000
8000	100	0.841945	2000
10000	80	0.781593	1250
20000	40	0.620350	1000
40000	20	0.492373	750
60000	13	0.426512	500

Tabulka 8.1: Parametry použité při simulacích pro zjištění časové a paměťové složitosti. n je počet částic, m hmotnost jedné částice, h je velikost okolí a k je konstanta tuhosti.

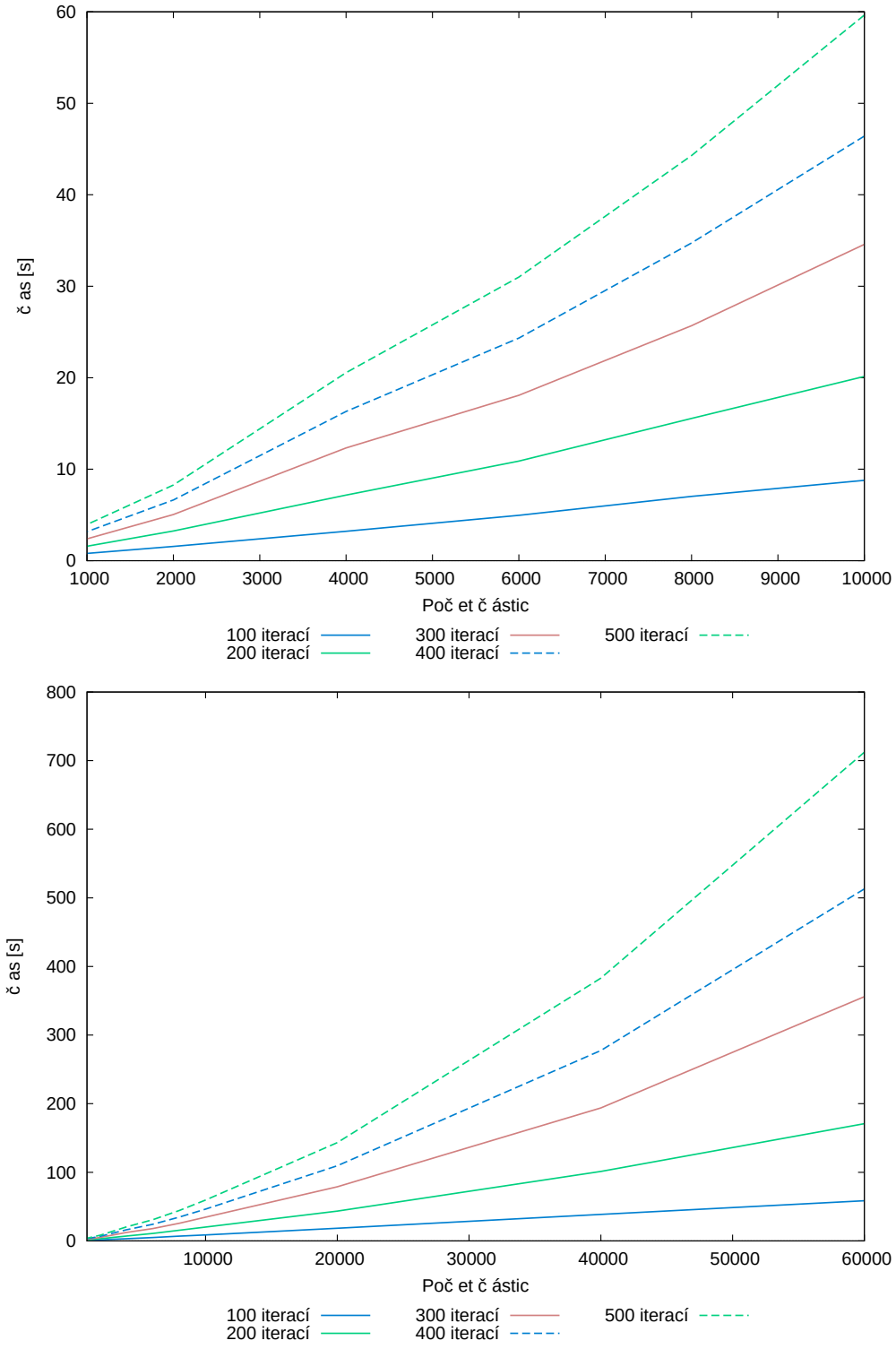
n	100 it.	200 it.	300 it.	400 it.	500 it.
1000	17.63	22.35	16.46	16.74	17.67
2000	19.94	23.85	17.96	18.20	18.82
4000	20.43	30.85	30.77	17.99	21.02
6000	19.87	25.84	22.98	19.72	20.39
8000	20.94	27.02	22.07	20.07	20.87
10000	21.01	33.06	24.88	20.43	21.74
20000	21.48	35.56	32.22	21.06	22.20
40000	22.51	47.42	28.42	23.48	23.10
60000	22.67	60.62	35.50	22.93	26.44

Tabulka 8.2: Průměrný počet sousedů.

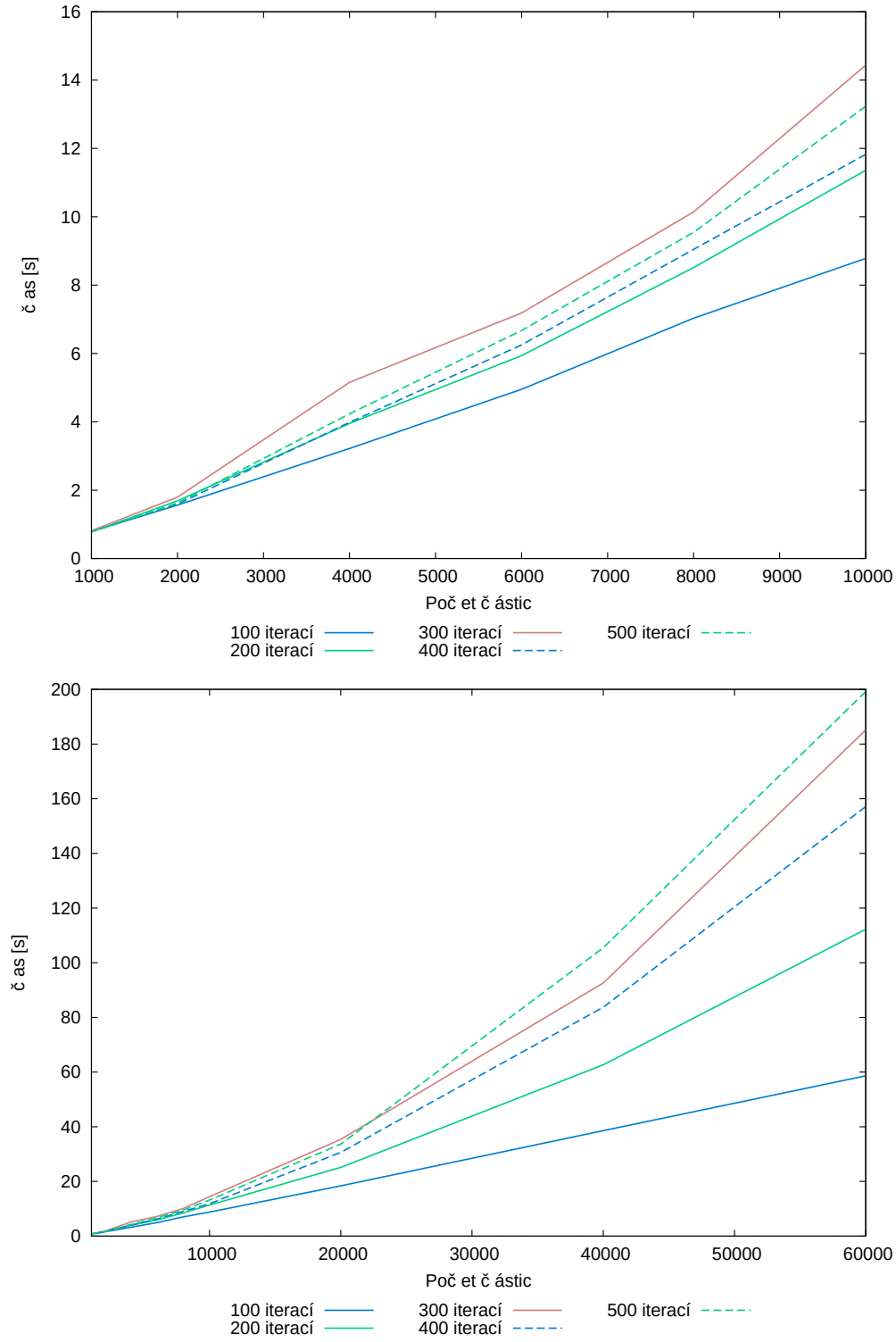
8.4 Časová a paměťová složitost

Na obrázku 8.5 jsou grafy, ve kterých jsou vyneseny časové závislosti simulace pro různý počet částic (na horním grafu pro rozsah $1000 \ldots 10000$ částic, na spodním $1000 \ldots 60000$) a pro $100 \ldots 500$ počet iterací. V testovací scéně (obrázek 8.8) byly částice vygenerovány v pravidelných rozestupech do tvaru hranolu. Prvních přibližně 100 iterací padá celý objem volným pádem, poté narazí na stěnu kontejneru. Z grafu je patrné, že čas potřebný pro prvních 100 iterací závisí přibližně lineárně na počtu částic v celém rozsahu testu. Pro více iterací už časová závislost není ani přibližně lineární. Toto zjištění je nepříjemné, protože z něj plyne, že rychlost simulace záleží nejenom na počtu částic, ale i na ostatních parametrech simulace. Na obrázku 8.6 jsou dva grafy s vynesenými časy pro posledních 100 iterací, jedná se tedy o rozdíly křivek z grafů na obrázku 8.5. Zde je dobře patrná závislost na stavu scény. Iterace $300 \ldots 400$ proběhnou rychleji, než iterace $200 \ldots 300$ nebo $400 \ldots 500$.
Příčin tohoto chování je několik. Při dynamičtějším chování simulovaných částic dochází častěji k přechodu částice do jiné buňky mřížky, která slouží k urychlení vyhledávání sousedů. Při těchto přechodech je třeba tuto mřížku aktualizovat, což jsou operace se složitostí $O(\log(n))$[6](kde n odpovídá počtu buněk mřížky) pro použitou datovou strukturu. Tyto aktualizace mají svůj podíl na časové složitosti. Dalším faktorem je průměrný počet sousedů částice. Velikost okolí (viz kapitola 7.2.5) je nastavena tak, aby počet částic v okolí byl konstantní. To ovšem platí pouze pro kapalinu v klidu, přesněji ve stavu kdy hustota ρ je rovna klidové hustotě ρ_0. Je-li hustota vyšší, pak i počet částic je vyšší. Tento jev má za následek nejen zpomalení výpočtu, neboť náročnost výpočtů závisí na počtu sousedů, má tedy složitost

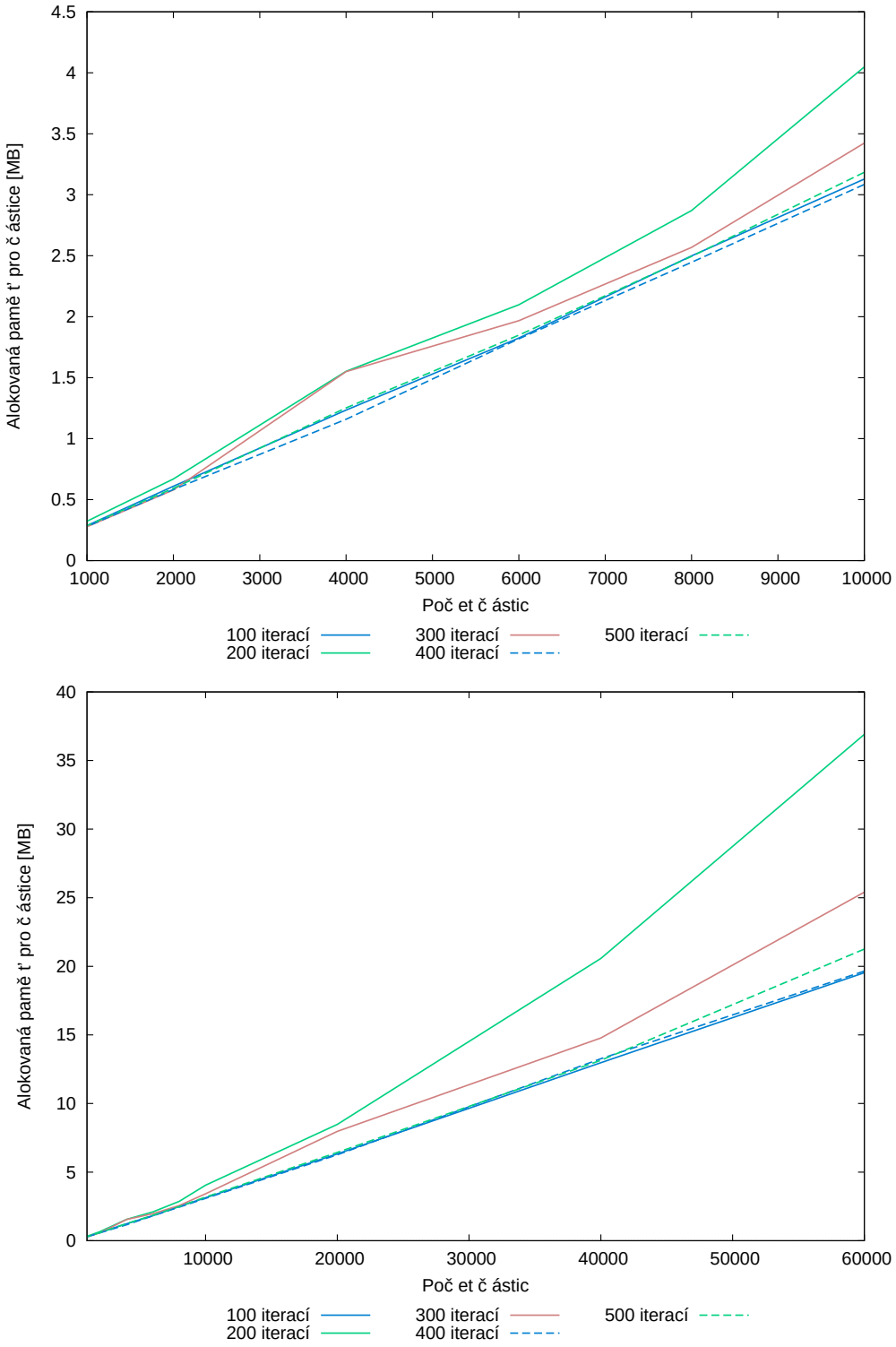
$O(ns_{avg})$ (n je počet částic, s_{avg} je průměrný počet sousedů), ale i paměťové nároky jak je patrné na obrázku 8.7, protože sousední částice jsou uchovávány v lineárním seznamu. Mezi grafy na obrázcích 8.7 a 8.6 lze proto sledovat jisté podobnosti. V grafech alokované paměti jsou udávány velikosti pouze dat uložených v datových strukturách, není započítána vlastní režie těchto struktur.
V tabulce 8.1 jsou uvedeny důležité parametry použité pro získání dat i pro vyhodnocení časové a paměťové složitosti. n je počet částic, m hmotnost jedné částice, h je velikost okolí a k je konstanta tuhosti. Je vidět, že paměťové nároky i rychlost simulace korespondují s parametrem k který určuje tuhost kapaliny. Čím je jeho hodnota menší, tím větších hustot může kapalina dosáhnout a tím je větší průměrný počet sousedů. Průměrný počet sousedů je uveden v tabulce 8.2. Parametr k nemohl zůstat konstantní z důvodu stability. O tomto problému pojednává kapitola 7.2.6



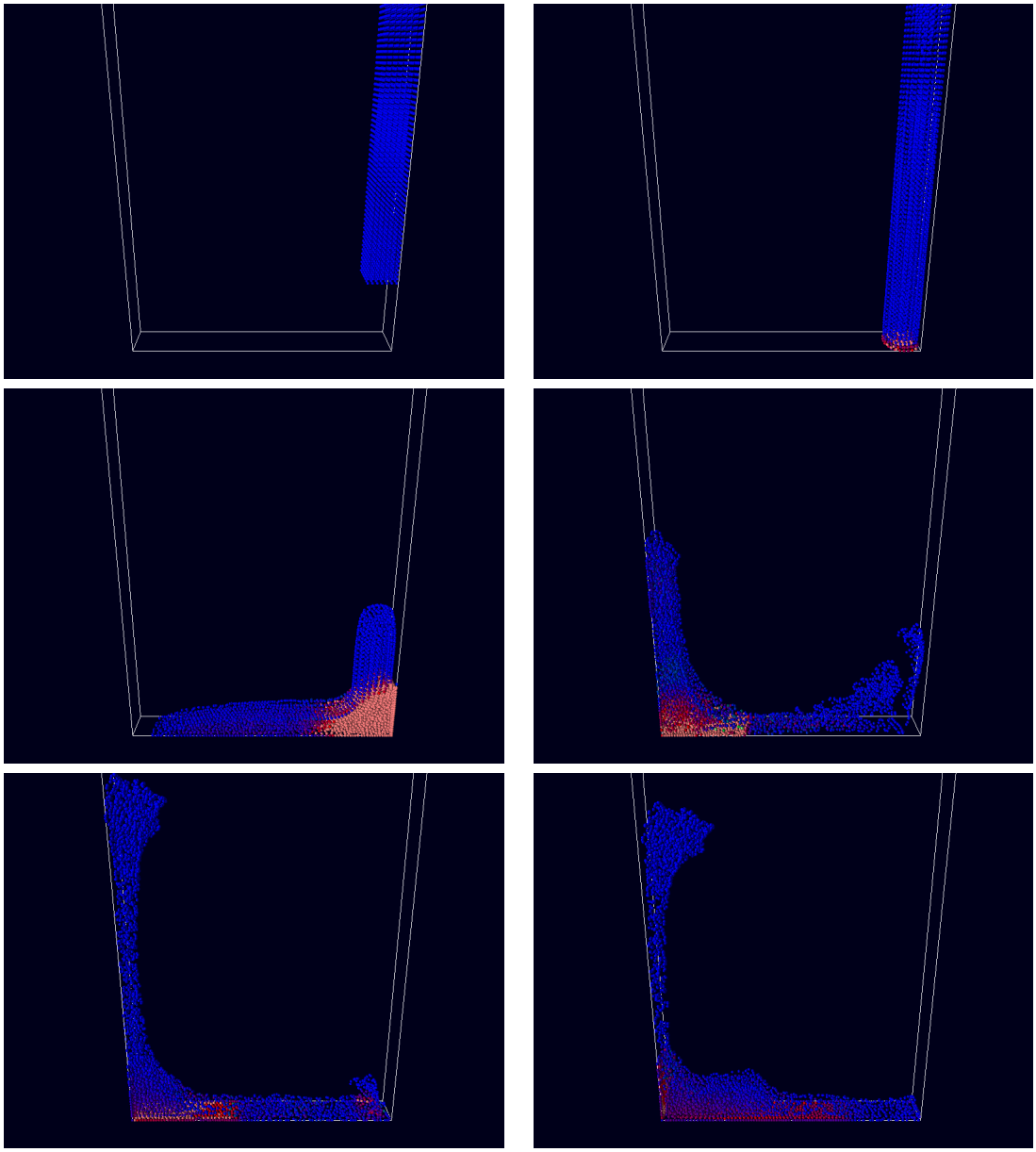
Obrázek 8.5: Časové závislosti pro 1000 . . . 10000 a 1000 . . . 60000 částic a 100 . . . 500 iterací simulační smyčky. Absolutní časy.



Obrázek 8.6: Časové závislosti pro 1000 . . . 10000 a 1000 . . . 60000 částic a 100 . . . 500 iterací simulační smyčky. Časy pro posledních 100 iterací.



Obrázek 8.7: Alokovaná paměť pro 1000 . . . 10000 a 1000 . . . 60000 částic a 100 . . . 500 iterací simulační smyčky.



Obrázek 8.8: Testovací scéna pro měření časové složitosti, snímky odpovídají stavu po 0, 100, 200, 300, 400 a 500 iteracích.

9 Závěr

V této práci jsme prošli cestu vedoucí k funkční simulaci viskózních kapalin počínaje výchozími Navier-Stokesovými rovnicemi přes SPH a aplikaci na Lagrangeovu metodu až po vlastní implementaci. Výsledná aplikace umožňuje simulaci kapalin pomocí až 3000 částic při zachování interaktivní rychlosti. Metoda SPH se ukázala být poměrně snadno implementovatelnou cestou k funkční simulaci kapalin.
Zřetelné jsou hlavní rozdíly mezi Eulerovým a Lagrangeovým přístupem. Prvně jmenovaný, ačkoli vychází ze stejných základních rovnic, řeší složitější soustavu rovnic, protože nemá zajištěno zachování hmoty tak jako systémy založené na částicích. Eulerovy metody jsou také obecně pomalejší a náročnější na paměť, ale v minulosti bylo věnováno mnoho úsilí na překonání těchto nevýhod a výsledky představené v kapitole 2 se vyznačují vysokým stupněm realističnosti.
Metoda SPH má jednu velikou nevýhodu, která souvisí s oblastí, pro kterou byla tato metoda vyvinuta. SPH je určeno pro stlačitelné tekutiny a nelze simulovat nestlačitelnost kapalin. Lze se pouze více či méně přiblížit nestlačitelnosti, což pro potřeby počítačové grafiky může být v některých aplikacích dostatečné. Dalším problémem simulací založených na částicích je nalezení povrchu kapaliny. Není neobvyklé, že při použití SPH nalezení povrchu trvá násobky času simulace.
Rychlost simulace je závislá na časovém kroku integrace, který je závislý na parametrech kapaliny. Všechny simulace použité v této práci probíhaly stabilně s časovým krokem $10ms$, ale pokud by byla požadována například menší stlačitelnost, časový krok by musel být menší, což by se projevilo negativně na rychlosti simulace.
Dalším problémem, který zasluhuje pozornost, je problematika přítoku a odtoku kapaliny. Toto je snadno řešitelné pro simulace založené na mřížce, ale pro částicové systémy jsem nenalezl uspokojivé řešení. Přítok musí být řešen vytvářením nových částic a odtok jejich odstraňováním. Jakmile ovšem dojde k vytvoření částice blízko jiné částice, hrozí vznik nestability, která může ovlivnit celou simulaci. Proto jsou v této práci použity pouze scény, kdy jsou všechny částice generovány před spuštěním simulace ve formě pravidelné mřížky.

9.1 Budoucí práce

Výsledek této práce může sloužit jako základ dalšího rozšíření jak simulační části tak i části vizualizační. Díky architektuře, která striktně odděluje simulaci od vizualizace, není problém přidat další vizualizační techniky, včetně off-line vizualizace například pomocí metody sledování paprsku. Pokud v simulaci přibudou další stavové proměnné, není problém tyto veličiny přidat do vizualizace.
Pro vizualizaci povrchu, by bylo možné použít metodu založenou na naplácávání (*point splatting*) podobně jako v práci [7]. Pro tuto metodu není potřeba vzorkování dat v rovnoměrné 3D mřížce jako u algoritmu pochodující kostky, kdy toto vzorkování je neúnosně časově náročné (viz tabulka 7.1).
Do vlastního simulačního modelu by bylo možno přidat podporu elastického chování popsaného v [2], podporu více nemísitelných kapalin, vedení tepla a závislost parametrů kapaliny na teplotě (hustota, viskozita atd.) popsaných v [8].
Dalšího urychlení by bylo možno dosáhnout reorganizací simulační smyčky, kdy výpočty by byly prováděny vždy pro jednu buňku a ne pro všechny částice najednou. To by snížilo paměťovou náročnost, kterou tvoří hlavně seznamy sousedních částic, které by se pak mohly vytvářet pouze pro částice v jedné buňce, ale také by to mohlo vést k další paralelizaci výpočtu.

10 Literatura

[1] AGEIA. Stránka výrobce fyzikálního akcelerátoru physx pro pc. http://www.ageia.com/.

[2] T. G. Goktekin, A. W. Bargteil, and J. F. O'Brien. A method for animating viscoelastic fluids. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 463–468, New York, NY, USA, 2004. ACM Press.

[3] GPGPU. Stránka věnovaná obecným výpočtům na grafických kartách. http://www.gpgpu.org/.

[4] M. Kelager. Lagrangian fluid dynamics using smoothed particle hydrodynamics. Master's thesis, Department of Computer Science University of Copenhagen, 2006.

[5] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 457–462, New York, NY, USA, 2004. ACM Press.

[6] Microsoft. Standard c++ library reference.

[7] M. Müller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 154–159, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[8] M. Müller, B. Solenthaler, R. Keiser, and M. Gross. Particle-based fluid-fluid interaction. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 237–244, New York, NY, USA, 2005. ACM Press.

[9] OpenGL. Stránka věnovaná opengl, multiplatformnímu rozhraní pro počítačovou grafiku. http://www.opengl.org/.

[10] S. Premoze, T. Tasdizen, J. Bigler, A. Lefohn, and R. T. Whitaker. Particle-based simulation of fluids. In *Proceedings of Eurographics 2003*, pages 401–410, 2003.

[11] A. Selle, N. Rasmussen, and R. Fedkiw. A vortex particle method for smoke, water and explosions. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 910–914, New York, NY, USA, 2005. ACM Press.

[12] D. Stora, P.-O. Agliati, M.-P. Cani, F. Neyret, and J.-D. Gascuel. Animating lava flows. In *Graphics Interface (GI'99) Proceedings*, pages 203–210, Jun 1999.

[13] M. Vesterlund. Simulation and rendering of a viscous fluid using smoothed particle hydrodynamics. Master's thesis, Department of Computer Science University of Umea, 2004.

[14] R. Vignjevic. Review of development of the smooth particle hydrodynamics (sph) method. Technical report, Cranfield University; School of Engineering, 2004.

[15] Wikipedie. Otevřená encyklopedie. http://www.wikipedia.org/.

A Ovládání aplikace

Aplikace je tvořena jedním spustitelným souborem `Flow.exe` a dvěmi dynamickými knihovnami `Renderer.dll` a `Simulation.dll`. Při spuštění souboru `Flow.exe` je jako parametr vyžadován konfigurační soubor. Základním prostředkem pro ovládání aplikace je konzole zobrazitelná klávesou `~`, několik klávesových zkratek a myš.
Klávesové zkratky obecně fungují, pokud je konzole schovaná. Při zobrazení konzole fungují pouze `ESC` a `F11`

- `w, a, s, d` nebo šipky - pohyb pozorovatele.
- `0 - 9` - přepnutí do vizualizačního režimu 0-9.
- `+, -` - přepnutí do následujícího/předchozího vizualizačního režimu.
- `F11` - uložení snímku obrazovky.
- `ESC` - ukončení aplikace.
- mezerník - jeden krok simulace.
- `r` - spuštění simulace.

Při stisknutém levém tlačítku myši a pohybu dochází k otáčení pozorovatele. Kurzor myši vybírá částici, pokud je povolen výběr částice parametrem `showselection`. Stisk levého tlačítka myši při současném stisku klávesy `Ctrl` uzamkne výběr na částici pod kurzorem.

B Struktura konfiguračního souboru

Konfigurační soubor je textový soubor, který je čtený po řádcích. Text na řádce po znaku `#` je považován za komentář. Prázdné řádky jsou ignorovány. Každý řádek poté definuje jeden parametr a má tento formát:

⟨jméno parametru⟩⟨typ⟩⟨hodnota⟩,

kde typ může nabývat následujících hodnot

- `int` celočíselný parametr, hodnota je celé číslo, odpovídá typu `int` jazyka C++.
- `double` reálný parametr, hodnota je reálné číslo, odpovídá typu `double` jazyka C++.
- `bool` booleovský parametr, jako hodnotu je možno použít konstanty *true* a *false*, nebo číselné hodnoty 0 a 1, kdy 1 odpovídá konstantě *true*. Odpovídá typu `bool` jazyka C++.
- `vector` trísložkový vektor reálných čísel. Hodnota je zapsána ve formě tří mezerou oddělených reálných čísel. Odpovídá typu `C3DVector`.
- `string` řetězec znaků, odpovídá typu `std::string`.

C Seznam parametrů a příkazů

Všechny příkazy a několik parametrů je definováno staticky přímo ve zdrojovém kódu. Zbylá většina parametrů je vytvořena dynamicky načtením z konfiguračního souboru. K příkazům a parametrům je přístup přes konzoli programu. Hodnota parametru je vypsána po zapsání jeho jména do konzole, pokud za jméno připojíme i hodnotu, je tato do parametru uložena. Příkaz se vyvolá podobně a některé příkazy přijímají i hodnoty jako své parametry.

Jméno	typ	popis
help	příkaz	Vypíše seznam příkazů a parametrů. Pokud je řetězec parametru příkazu neprázdný, vypíše pouze ty příkazy a parametry začínající tímto řetězcem.
r_update	příkaz	Obnoví datové struktury používané pro vizualizaci.
osd	příkaz	Umožní přidávání hodnot do informací zobrazovaných v levém horním rohu okna. Formát parametru příkazu je ⟨formátovací řetězec⟩⟨jméno parametru kde %% ve formátovacím řetězci bude nahrazeno hodnotou parametru.
r_mode_name	příkaz	Slouží ke zjištění jména konkrétního vizualizačního módu.
hm_init	příkaz	Načte výškovou mapu.
sim_init	příkaz	Zinicializuje částice.
sim_density	průměr	Parametr který uchovává průměrnou hustotu částic.
sim_neighbours	průměr	Parametr který uchovává průměrný počet sousedů částice.
sim_t	double	Simulační čas.
Následující parametry jsou definovány v konfiguračním souboru.		
sim_mode	int	Parametr kontrolující běh simulace, pro hodnotu 0 simulace běží bez přerušení, pro hodnotu 1 simulace pokročí o jeden časový krok, pro jiné hodnoty se simulace zastaví.
wireframe	bool	Definuje zda se má vykreslovat pouze drátěný model.
lightning	bool	Definuje zda má být použito osvětlení.
shownormals	bool	Definuje zda se mají zobrazit normály výškové mapy.
showosd	bool	Definuje zda se mají zobrazit textové informace v levém horním okraji obrazovky.
showgauge	bool	Definuje zda se má zobrazit legenda.
showselection	bool	Definuje zda má probíhat výběr částice pomocí kurzoru.
move_size	double	Velikost kroku pozorovatele při pohybu.
rotate_size	double	Velikost kroku pozorovatele při otáčení.
light_dir	vector	Směr odkud dopadá světlo na scénu.
r_hm	bool	Definuje zda se má zobrazovat výšková mapa.
r_box	bool	Definuje zda se má zobrazovat kolizní hranol.
r_plane	bool	Definuje zda se má zobrazovat kolizní rovina.
cam_pos	vector	Umístění pozorovatele ve scéně.
cam_look	vector	Bod který pozorovatel sleduje.
r_fluid_colour	vector	Barva povrchu kapaliny. RGB složky.
r_fluid_alpha	double	Míra průhlednosti povrchu kapaliny. Rozsah 0 → 1, 1 neprůhledná, 0 zcela průhledná.
r_mode	int	Výběr veličiny která se má vizualizovat. Pro hodnotu 0 se vizualizuje povrch kapaliny.
r_mode_#_min	double	# zastupuje číslo veličiny. Pro skalární veličiny, předpokládaná minimální hodnota veličiny.

r_mode_#_max	double	# zastupuje číslo veličiny. Pro skalární veličiny, předpokládaná maximální hodnota veličiny.
r_mode_#_scale	double	# zastupuje číslo veličiny. Pro vektorové veličiny, konstanta pro obarvování vektorů.
sim_g	vector	Vektor gravitačního zrychlení.
sim_dt	double	Časový krok simulace v sekundách.
sim_cr	double	Míra elasticity srážek c_r.
sim_capture	bool	Určuje zda se bude výsledek vizualizace ukládat do videosekvence.
sim_fps	double	Počet snímků na vteřinu videosekvence.
capture_file	string	Jméno souboru s výslednou videosekvencí.
sim_stoptime	double	Čas v sekundách po kterém dojde k zastavení simulace. Pro hodnotu menší nebo rovnu nule k zastavení nedojde.
sim_colision	int	Typ kolizí. 0 - pouze kolize s hranolem, 1 - kolize s hranolem a rovinou, 2 - kolize s hranolem a výškovou mapou.
fullinit	bool	Určuje zda se mají při inicializaci vytvořit všechny částice.
init_pos	vector	Pozice hranolu ve kterém budou částice při inicializaci vytvořeny.
init_size_xy	double	Velikost hranolu ve kterém budou částice při inicializaci vytvořeny.
inflow	double	Pokud nebudou všechny částice vytvořeny při inicializaci, tento parametr určí kolik nových částic se vyvoří za sekundu.
inflow_vel	vector	Počáteční rychlost částic.
hm_name	string	Jméno souboru s výškovou mapou.
hm_size_xy	double	velikost výškové mapy v metrech.
hm_base_z	double	Z-ová souřadnice nejnižšího bodu výškové mapy.
hm_size_z	double	Rozsah výšek výškové mapy v metrech.
hm_samples	int	Počet vzorků výškové mapy pro vizualizaci.
box_size_x	double	Velikost kolizního hranolu ve směru lokální osy x.
box_size_y	double	Velikost kolizního hranolu ve směru lokální osy y.
box_size_z	double	Velikost kolizního hranolu ve směru lokální osy z.
box_pos	vector	Pozice kolizního hranolu.
box_up	vector	Orientace lokální osy z kolizního hranolu.
box_right	vector	Orientace lokální osy x kolizního hranolu.
box_move	vector	Rychlost posunu kolizního hranolu.
plane_pos	vector	Pozice kolizní roviny.
plane_normal	vector	Normála kolizní roviny.
particles	int	Počet částic.
p_stiff	double	Konstanta tuhosti kapaliny k.
p_visc	double	Konstanta viskozity kapaliny μ.
p_rest_dens	double	Klidová hustota kapaliny ρ_0.
p_mass	double	Hmotnost jedné částice v kilogramech.
p_surface_force_coef	double	Koeficient povrchového napětí σ
r_mc	bool	Určuje zda se bude provádět vzorkování potřebné pro metodu pochodující krychle. Je-li false nebude zobrazován povrch kapaliny.
r_mc_subdiv	int	Určuje jemnost mřížky kterou se vzorkuje prostor pro metodu pochodující krychle.

r_mc_iso	double	Tato hodnota rozděluje vzorky na vnitřní (s větší hodnotou) a vnější (s menší hodnotou).

D Obsah přiloženého CD

- *index.html* - rozcestník projektu
- *readme.txt* - informace o spuštění a ovládání programu, systémové nároky
- *install.bat* - instalační dávka, překopíruje soubory nutné pro běh do zvoleného adresáře
- *adresář bin/* - přeložený program `Flow.exe`, včetně knihoven; dávka `demo.bat` spustí program s vyžadovanými parametry
  - adresář *configs/* - konfigurační soubory použité pro generování videosekvencí a demonstračních scén použitých v práci
  - adresář *img/* - používané textury a výškové mapy
- adresář *html/*
  - adresář *abstrakt/* - abstrakt diplomové práce
  - adresář *implementace/* - dokumentace popisující implementaci programu
    - adresář *html/* - dokumentace ve formátu HTML, kořenový soubor je `index.html`
    - *Flow.chm* - dokumentace ve formátu Compiled HTML Help
  - adresář *RabstrCZ/* - rozšířený abstrakt v českém jazyce
  - adresář *RabstrEN/* - rozšířený abstrakt v anglickém jazyce
- adresář *src/* - zdrojové kódy programu v jazyce C++; k dispozici je též projekt pro Microsoft Visual Studio .NET 2003
- adresář *text/* - text diplomové práce
  - *dipl.pdf* - text diplomové práce ve formátu PDF; tento soubor byl ze zdrojových souborů vytvořen programem pdfLaTeX
  - adresář *src/* - zdrojové soubory textu včetně obrázků; hlavním souborem je `dipl.tex`
- adresář *video/* - videosekvence vytvořené programem; použitý kodek je Xvid[1]
  - *box_surf0.avi* - kapalina v prostředí bez gravitace bez započítání účinků povrchového napětí
  - *box_surf.avi* - stejné prostředí jako v předchozím případě, ale se započtením účinků povrchového napětí
  - *box_stiff_1000.avi* - kapalina s konstantou tuhosti $k = 1000$
  - *box_stiff_3000.avi* - kapalina s konstantou tuhosti $k = 3000$
  - *box_visc_1000.avi* - kapalina s viskozitou $\mu = 1000$
  - *box_visc_5000.avi* - kapalina s viskozitou $\mu = 5000$
  - *test_hm.avi* - tok kapaliny prostředím definovaném výškovou mapou

[1] http://www.xvid.org