VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ČÁSTICOVÉ SIMULACE V REÁLNÉM ČASE

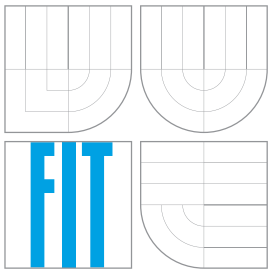
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ZSOLT HORVÁTH

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ČÁSTICOVÉ SIMULACE V REÁLNÉM ČASE

REAL-TIME PARTICLE SIMULATIONS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ZSOLT HORVÁTH

VEDOUCÍ PRÁCE
SUPERVISOR

doc. Ing. ADAM HEROUT, Ph.D.

BRNO 2012

Abstrakt

Částicové simulace v reálném čase se stali realitou teprve před několika roky, kdy se v informatice objevil pojem GPGPU. Tato nová technologie umožňuje využívat obrovskou sílu grafické karty pro obecné účely. V dnešní době je trendem urychlit existující algoritmy přepsáním do paralelní podoby. Na tomto principu fungují i částicové systémy. Zajímavou oblastí částicových systémů jsou simulace tekutin. Tyto simulace jsou založené na teorii Navier-Stokesových rovnic a jejich numerickém řešení pomocí SPH (Smoothed particle hydrodynamics). Tekutiny tvoří součást každodenního života a proto je důležité je zobrazit realisticky. Používají se v moderních počítačových hrách a v různých vizualizacích, které běží v reálném čase, z toho důvodu musí být rychle zobrazené.

Abstract

Particle simulations in real-time become reality only a few years before, when in computer science occured the idea of GPGPU. This new technology allows use the massive force of graphics card for general purposes. Today, the trend is to accelerate existing algorithms by rewriting into parallel form. On this priciple operate the particle systems too. An interesting area of particle systems are fluid simulations. The simulations are based on the theory of Navier-Stokes equations and their numerical solutions with SPH (Smoothed particle hydrodynamics). Liquids are part of everyday life, and therefore it is important to render them realistically. They are used in modern computer games and different visualizations that run in real time, therefore they must be quickly displayed.

Klíčová slova

Částicové simulace, částicové systémy, SPH (Smoothed Particle Hydrodynamics), CUDA, GPGPU, Marching cubes, teselace, Navier-Stokes, OpenGL, SDL, simulace tekutin

Keywords

Particle simulations, particle systems, SPH (Smoothed Particle Hydrodynamics), CUDA, GPGPU, Marching cubes, tessalation, OpenGL, SDL, fluid simulations

Citace

Zsolt Horváth: Částicové simulace v reálném čase, diplomová práce, Brno, FIT VUT v Brně, 2012

Částicové simulace v reálném čase

Prohlášení

Prohlašuji, že jsem tuto semestrálnu práci vypracoval samostatně pod vedením pana Doc. Ing. Adama Herouta, Ph.D.

. . . . . .
Zsolt Horváth
13. května 2012

Poděkování

Chtěl bych se poděkovat vedoucímu práce, Doc. Ing. Adamovi Heroutovi, Ph.D., za poskytnutí pomoci a podkladů a za podporu, kterou projevoval po celou dobu vytvoření této práce.

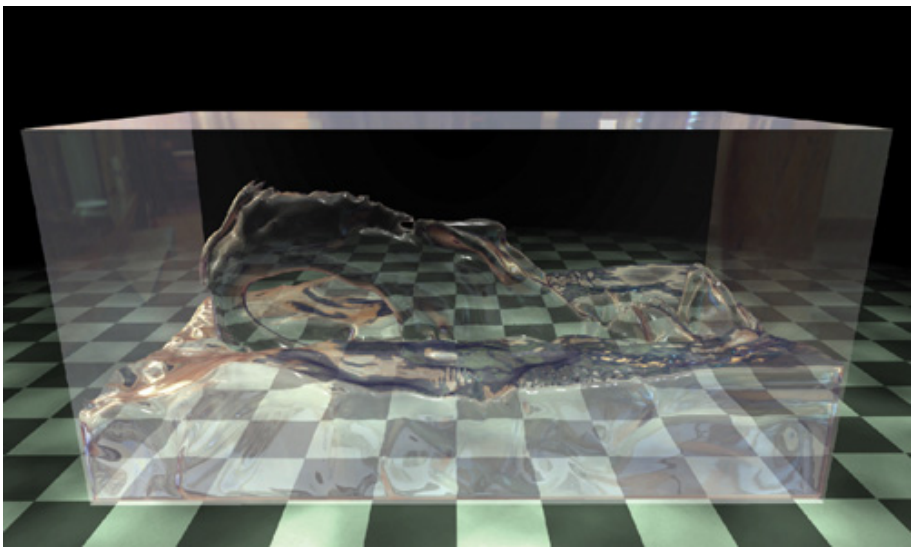
Obsah

Kapitola 1

Úvod

Cieľom tejto práce bolo vytvoriť časticový systém [25] [14], ktorý bude simulovať kvapaliny v reálnom čase s využitím platformy CUDA. Simulačný systém je založený na Langeovej metóde. V simulácii sa používa fyzikálny model nestlačiteľnej tekutiny, ktorý je popísaný rovnicami Navier-Stokesa. Tieto rovnice sú implementované pomocou metódy SPH.

V počítačovej grafike, ktorá funguje v reálnom čase sa snažíme vždy reprodukovať časť sveta tak, aby bola čo najrealistickejšia. Bohužiaľ je ťažké simulovať tieto javy kvôli obmedzeným výkonom počítačov. Väčšina simulátorov funguje *offline*. To znamená, že výpočty nie sú dostatočne rýchle na to, aby sa dali výsledky zobrazovať v reálnom čase, preto sa generujú postupne. Veľký nedostatok týchto simulátorov je, že neumožnujú užívateľovi interakciu. Sila moderných grafických kariet už začína prekonávať tieto hranice. Dnes už môžeme napísať programy a spusiť ich na grafických kartách, ktoré majú veľký počet výkonných jadier a umožňujú vypočítať všeobecné úlohy. Táto technológia je výrazne rýchlejšia ako výpočet v CPU.



Obrázek 1.1: Časticový systém simulujúce tekutinu z [18].

S kvapalinami sa stretávame každý deň. Keď pijeme vodu, umývame sa alebo napríklad keď, vonku prší. Sú všade okolo nás. Preto je dôležité pochopiť, ako sa správajú, ako vyzerajú. Ich dynamické chovanie je popísané differenciálnými rovnicami. Simulácia kvapalín

patrí medzi ťažšie úlohy časticových simulácií. Systém simulujúci granulárne materiály, ako je napríklad piesok, sa modeluje jednoduchšie. U týchto typov nie je potrebné riešiť zložité sady differenciálných rovníc.

Kapitola 2

Simulácie kvapalín

V tejto kapitole bude podrobnejšie popísaná teória simulácie kvapalín a platformy CUDA.

2.1 Časticové simulácie

Keď zobrazujeme objekty v počítačovej grafike, tak potrebujeme ich tvar, povrch. K popisu tvaru sa používajú poligoniálne siete. Fyzikálne animácie takýchto sietí fungujú len pre objekty, ktoré majú presne definovaný tvar, ako napríklad nábytok či šaty. S *fuzzy* objektmi sú problémy. *Fuzzy* objekt môže byť voda, ohňostroj, mraky, dym či požiar. Objekty takého typu je ťažké modelovať s poligoniálnymi sieťami, lebo majú dynamicky meniteľný tvar a hladký povrch. K simulovaniu takýchto objektov musíme poznať objem namiesto povrchu. Podľa prístupu volumetrické simulácie delíme typicky na dve metódy: *Eulerove* a *Lagrangeove*. V *Eulerovej* máme vopred definovanú mriežku v trojdimenzionálnom priestore, vypočítame zmeny tlaku a rýchlosti v jednotlivých bodoch danej mriežky. *Lagrangeova* metóda určuje vlastnosti jednotlivých častíc. Tu nie je potrebná mriežka, častice sa pohybujú spolu s kvapalinou.

Koncept častíc, alebo časticových systémov prvýkrát predstavil Reeves [30], aby simuloval *fuzzy* objekty v počítačovej grafike. Deformáciu trojuholníkovej siete kvapaliny nie je jednoduché riešiť, ale deformovať častice a následne extrahovať plohu je oveľa jednoduchšie.

Parametre častíc

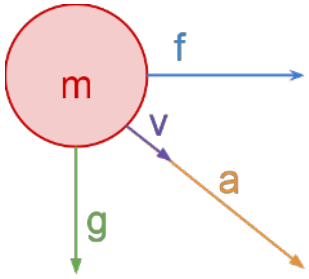
Častice si pamätajú svoje parametre, ktoré popisujú ich aktuálny stav. Tieto parametre sú pozícia (p), hmota (m), objem (V) a hustota (ρ). Môžu samozrejme sa pohybovať v priestore s určitou rýchlosťou (v) v závislosti na aktuálnom zrýchlení (a).

Dynamika častíc

Jednotlivé častice môžu meniť svoje vlastnosti počas života. Použitím *Newtonovho* druhého zákona [31] zrýchlenie (a) závisí od vonkajších síl (f) a od gravitačného zrýchlenia (g).

$$a = \frac{\mathrm{d}v}{\mathrm{d}t} = \frac{f}{m} + g \tag{2.1}$$

Vonkajšie sily je jednoduché vypočítať ak poznáme gravitačné zrýchlenie a hmotu (m) častice. Z týchto parametrov máme zrýchlenie častice. Stačí už len numericky integrovať vypočítanú hodnotu podľa času a dostaneme aj rýchlosť (v) častice. Už je len treba aktualizovať pozíciu, čo je triviálne.



Obrázek 2.1: Dynamika častíc.

2.2 Rovnice Navier-Stokes

Rovnice Navier-Stokes sú základom simulácie kvapaliny. Sú to differenciálne rovnice, ktoré sú riešiteľné analiticky len v jednoduchých pripadoch. Obecne sa riešia numerickými metódami. Rovnice popisujú chovanie nestlačiteľnej newtonovskej kvapaliny. Dôležitou vlastnosťou takej kvapaliny je, že dynamická viskozita je konštanta úmernosti medzi napätím a rýchlosťou deformácie.

$$\tau = -\eta \frac{\mathrm{d}v}{\mathrm{d}t} \tag{2.2}$$

τ je dotykové napätie v tekutine, u je rýchlosť toku a η je dynamická viskozita. Pre také kvapaliny dostaneme Navier-Stokeseve rovnice

$$\rho\left(\frac{\partial v}{\partial t} + v \cdot \nabla v\right) = -\nabla p + \mu \nabla^2 v + f \tag{2.3}$$

$$v \cdot \nabla = 0, \tag{2.4}$$

kde ρ je hustota kvapaliny, v je vektor rýchlosti, $\cdot$ je operácia skalárneho súčinu, ∇ označuje gradient, p je tlak, μ je viskozita, f je vektor externých síl. Rovnice vychádzajú zo zákona zachovania energie pre kvapaliny. Na ľavej strane je súčin hustoty a zrýchlenia. Zrýchlenie sa skladá z dvoch častí, nerovnomerná $\frac{\partial v}{\partial t}$ závisiaca na čase a konvektívna $v \cdot \nabla v$. Konvektívna časť v sebe zahrňuje zrýchlenie spôsobené tlakovým spádom (gradientom), zrýchlenie z objemových síl a zrýchlenie potrebné k prekonaniu trecích síl.

2.3 Metódy Eulera a Langranga

Riešenie rovníc *Navier-Stokes* môže byť časovo veľmi náročné, najmä pre 3-rozmernú reprezentáciu. Dôvodom je, že numerické riešenie obvykle vedie k nelineárnym parciálnym diferenciálnym rovniciam. Tie môžu byť riešené na základe rozdelenia objemu do miežky. Tieto metódy sú zvyčajne označované ako eulerovské metódy.

Alternatívou metódy *Euler* je metóda *Lagrange* (časticovo založená). Táto metóda pracuje tak, že tekutinu rozdelí na jemné častice a aplikuje rovnice kvapalín z časticovej mechaniky.

V podstate Eulerovská metóda vychádza z fixného objemu, cez ktorý tečie tekutina. Z Lagrangeovej formulácie vyplýva, že sleduje individuálne častice pohybujúce sa cez priestor v čase.

Eulerova formulácia

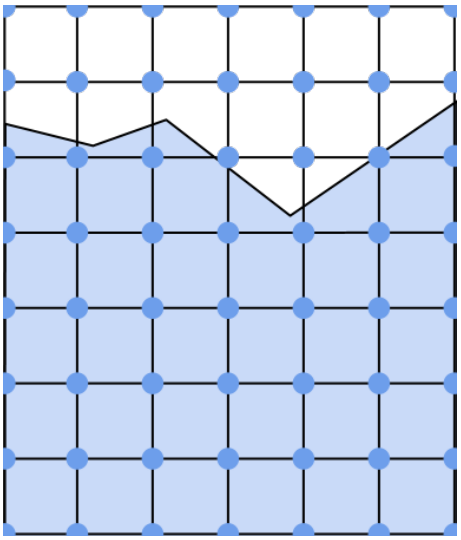
Rovnica popisujúca zachovanie hmoty / kontinuity je daná vzťahom:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \tag{2.5}$$

Zachovanie momentu je daná rovnicou:

$$\rho \left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} + v \ \cdot \ \nabla \mathbf{v} \right) = -\nabla p + \nabla \ \cdot \ \tau + \mathbf{f}, \tag{2.6}$$

kde $\mathbf{v}$ je rýchlostné pole, ρ je hustota, P je tlak, τ je viskózne napätie. Časť advekcie ($v \ \cdot \ \nabla \mathbf{v}$) v *Navier-Stokesovej* rovnici môžeme popísať ako časovo nezávislé zrýchlenie tekutiny vzhľadom na priestor. V podstate kvapalina je transportovaná po vlastnom toku.



Obrázek 2.2: Štruktúra tekutiny poďla Eulerovej formulácie v 2D. Vlastnosti tekutiny sú vypočítané v bodoch mriežky.

Lagrangeova formulácia

Rovnica popisujúca zachovanie hmoty / kontinuity je daná vzťahom:

$$\frac{d\rho}{dt} = -\rho \nabla \cdot \mathbf{v} \tag{2.7}$$

Použitím derivátu, ktorý je špecifikovaný takto: $\frac{d\mathbf{v}}{dt} = \frac{\partial \mathbf{v}}{\partial t} + v \ \cdot \ \nabla v$, dostaneme Lagrangeovu formuláciu zachovania momentu:

$$\frac{d\mathbf{v}}{dt} = -\frac{1}{\rho} \nabla p + \frac{1}{\rho} \nabla \ \cdot \ \tau + \mathbf{f} \tag{2.8}$$

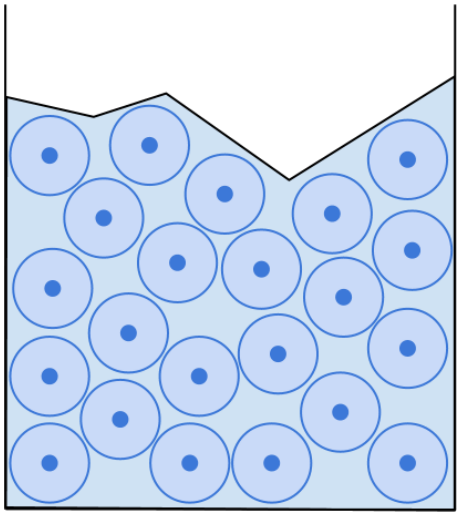
Môžeme ignorovať zachovanie hmoty, ak budeme predpokladať, že všetky častice majú rovnakú hmotu a máme konštatný počet častíc.

Nakoniec dostaneme výraz:

$$a_i = -\frac{1}{\rho_i} \nabla p_i + \frac{1}{\rho_i} \nabla \ . \ \tau_i + \mathbf{f}_i = f_i^{pressure} + f_i^{stress} + f_i^{external} \tag{2.9}$$

kde a_i je zrýchlenie častice, $f_i^{pressure}$ je tlaková sila, f_i^{stress} je napätie spôsobené viskozitou a $f_i^{external}$ je suma externých síl (napr. gravitácia, odrážanie od steny).

Ostatné rovnice sú jednoducho odvoditeľné z predchádzajúcich, alebo je možné nájsť napr. v [23] a [21].



Obrázek 2.3: Štruktúra časticovej tekutiny poďla Lagrangeovej formulácie v 2D. Častice sú reprezentované bodmi. Kruhy predstavujú objem jednotlivých častíc.

2.4 Smoothed particle hydrodynamics

Smoothed particle hydrodynamics je metóda k aproximácie riešenia numerických riešení rovníc dynamiky tekutín. Základnou myšlienkou je reprezentovať objem vzorkovaný metódou Monte Carlo vyhladených interagujúcich častíc. Je to možné tým, že reprezentujeme objem ako množinu prvkov (častíc) a používame tieto častice ako intrepolačné body, pre ktoré je možné vypočítať vlastnosti tekutiny. Metóda SPH bola pôvodne vyvinutá pre astrofyziku, ale dnes sa používa na radu problémov. Deriváciou sily viskozity a tlaku priamo v rovnici *Navier-Stokes* je možné veľmi efektívne simulovať chovanie tekutín s rôznymi hustotami.

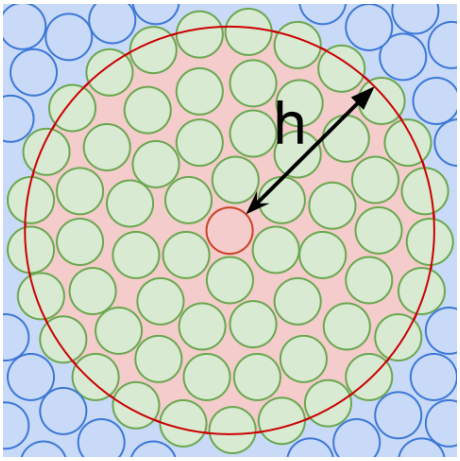
Metoda má niekoľko zaujímavých vlastností v kontexte simulácie v reálnom čase. Jedna z nich je, že môžeme jednoducho paralelizovať, čo značne urýchli dobu trvania výpočtov. Ďalšia výhoda je triviálne zachovanie hmoty, vysoká účinnosť a jednoducho sledovateľný povrch. Interakcia s komplexnými hranicami pevných látok je tiež omnoho jednoduchšia ako s tradičnými eulerovskými simuláciami, pretože nie je potrebné vypočítať zložité miežky.

Najväčším nedostatkom je to, že môže byť ťažké, alebo výpočetne náročné extrahovať jemný povrch z častíc. Táto metóda môže vyžadovať veľký počet častíc, aby dosiahla realistické výsledky.

V literatúre [26] je dostupný dobre napísaný článok o použití SPH orientovaný na enviromentálne vedy.

2.5 Rovnice SPH

V metóde SPH sú simulované rôzne efekty *Navier-Stokesových* rovníc ako množina síl, ktoré pôsobia na každú časticu. Tieto sily sú dané skalárnymi veličinami. Veličiny interpolujeme v mieste $\mathbf{r}$ na základe váženého súčtu príspevkov od všetkých okolitých častíc vo vzdialenosti h v priestore Ω (Obr. 2.4). Integrálom to môžeme napísať nasledovne:



Obrázek 2.4: Vyhladzovacia vzdialenosť a častice v nej.

$$A_i = \int_{\Omega} A\left(\mathbf{r}^{'}\right) W\left(\mathbf{r} - \mathbf{r}^{'}, h\right) \mathrm{d}\mathbf{r}^{'} \tag{2.10}$$

Numerický ekvivalent získame aproximáciou integrálu so sumou:

$$A_i = \sum_j A_j V_j W\left(\mathbf{r}_{ij}, h\right) \tag{2.11}$$

kde j je index susednej častice, V_j je implicitný objem častice j, $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$, kde $\mathbf{r}$ je pozícia častice a nakoniec A je skalárna interpolovaná veličina. Do sumy sa sčítajú častice, ktoré spadajú do kruhu stredom $\mathbf{r}_i$.

Medzi veľkosťou, hmotou a hustotou hmoty platí nasledujúci vzťah:

$$V = \frac{m}{\rho} \tag{2.12}$$

kde m je hmota, a ρ je hustota hmoty. Kombináciou tejto rovnice dostaneme základ formulácie interpolačnej funkcie SPH.

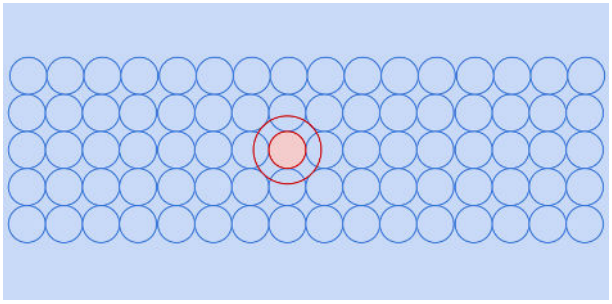
Nakoniec dostaneme základnú formuláciu interpolačnej funkcie SPH:

$$A_i = \sum_j A_j \frac{m_j}{\rho_j} W\left(\mathbf{r}_{ij}, h\right) \tag{2.13}$$

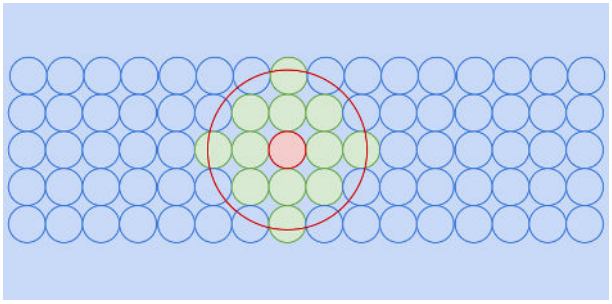
kde j je index susednej častice, m_j je hmota j-tého častice, r_j jej pozícia, ρ hustota a A_j je skalárna veličina v pozície r_j.

Vyhladzovacie jadro

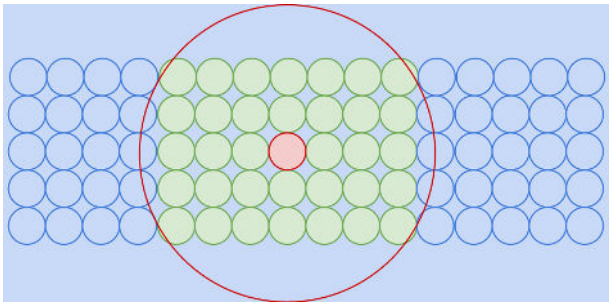
Funkcia $W(r_{ij})$ sa nazýva vyhladzovacie jadro, ktoré je vážená skalárna funkcia. Toto jadro používa pozíciu $\mathbf{r}$ a vyhladzovaciu dĺžku h. Na túto dĺžku môžeme pozerať ako na hodnotu prahu, ktorá rozhoduje o tom, koľko častíc sa započítava od interpolácie. Pre $|\mathbf{r}_{ij}| > h$ platí pre funkciu vždy $W = 0$. Na obrázku 2.5 sú naznačené rôzne veľkosti (krátka, ideálna a dlhá) vyhladzovacích jadier.



(a) Krátka.



(b) Ideálna.



(c) Veľká.

Obrázek 2.5: Vyhladzovacie dĺžky jadier.

Vyhladzovacie jadro musí spĺňať niekoľko podmienok. Prvá je normalizačná podmienka:

$$\int_r W(\mathbf{r}, h)\, \mathrm{d}r = 1 \tag{2.14}$$

Ďalšia podmienka je vlastnosť Dirakovej funkcie, ktorú môžeme pozorovať, keď vyhladzovacia dĺžka sa blíži k nule:

$$\lim_{x \to \infty} W(\mathbf{r}_{ij}, h) = \delta(\mathbf{r}_{ij}) \tag{2.15}$$

kde δ je Dirakova delta funkcia. Vysvetlenie Dirakovej delta funkcie je napr. v [13]. Tretia podmienka, ktorá zabezpečuje, že iba vnútorné častice sú používané pri výpočtoch, je nasledujúca:

$$W = 0 \; ak \; |\mathbf{r}_{ij}| > h \tag{2.16}$$

Okrem týchto podmienok ďalej musí platiť, že jadro je pozitívne a párne $W(\mathbf{r}, h) = W(-\mathbf{r}, h)$.

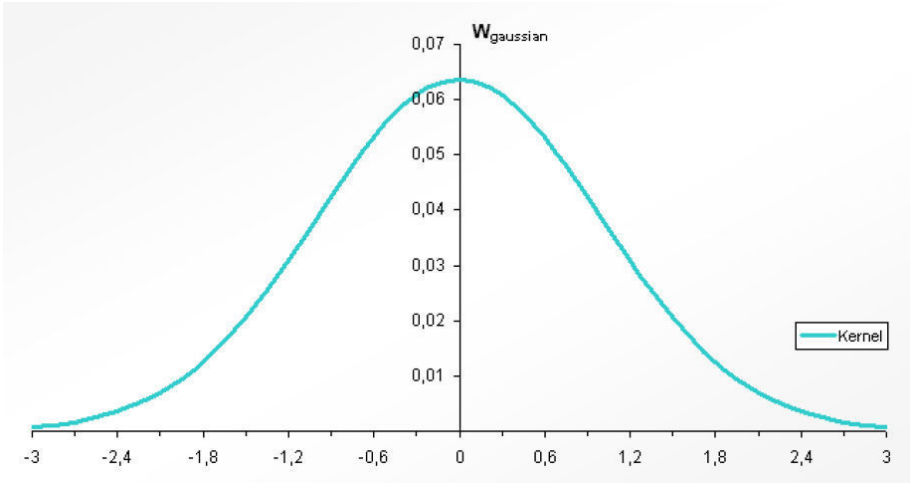
Ak W je delta funkcia, tak interpolačná funkcia bude produkovať veľmi presné výsledky. Avšak v praxi to nie je príliš užitočné, pretože vyžaduje veľkú vyhladzovaciu dĺžku a veľa susedných častíc. Tieto podmienky sú časovo náročné, preto aby sme zvýšili výpočený výkon, funkcia W si kladie za cieľ poskytovať čo najpresnejšiu interpoláciu, čo je možné pri zachovaní rozumnej vyhladzovacej dĺžky.

V [22] a [20] sú zavedené jadrá pre rôzne oblasti efektov, ako sú viskozita a tlak. Pomocou týchto jadier rôznych efektov je možné každé jadro optimalizovať pre špecifické požiadavky síl, ktoré interpolujeme.

Ak chceme pochopiť fyzikálnu interpretáciu rovnice SPH, je vždy najlepšie predpokladať, že jadro je gaussovské. To je prvé zlaté pravidlo SPH [17]. Gaussovské jadro je dané vzťahom:

$$W_{gaussian}(\mathbf{r}, h) = \frac{1}{(2\pi h^2)^{\frac{3}{2}}} e^{-\left(\frac{|\mathbf{r}|^2}{2h^2}\right)}, h > 0 \tag{2.17}$$

ktoré je vidieť na obrázku 2.6. Aj keď má gaussovské jadro veľmi pekné matematické vlast-



Obrázek 2.6: Gaussovské jadro v 1D, kde $h = 1$ z [21].

nosti, nie je vždy to najlepšie jadro na použitie, napr. vyžaduje vypočítanie drahých exponenciálov.

Deriváty

V SPH môžeme získať deriváty funkcií s použitím derivátov vyhladzovacieho jadra, čo vedie k *Basic Gradient Approximation Formula* (BGAF) [10].

$$\nabla A_i = \sum_j A_j \frac{m_j}{\rho_j} \nabla W(\mathbf{r}_{ij}, h) \tag{2.18}$$

$$\nabla^2 A_i = \sum_j A_j \frac{m_j}{\rho_j} \nabla^2 W(\mathbf{r}_{ij}, h) \tag{2.19}$$

Táto formulácia môže produkovať bohužiaľ nepresné výsledky, preto zaviedli niekoľko formulácií kvôli upresneniu výpočtu. Jedna z nich je *Difference Gradient Approximation Formula* (DGAF) [10]:

$$\nabla A_i = \frac{1}{\rho_i} \sum_j m_j \left(A_j - A_i\right) \nabla W\left(\mathbf{r}_{ij}, h\right) \tag{2.20}$$

ktorá má tú výhodu, že sila zmizne úplne, keď tlak je konštantný [16].

Symetrizácia gradientu

Pre sily, ktoré pôsobia medzi časticami platí tretí zákon Newtona, teda platí, že dva hmotné body na seba pôsobia rovnako veľkými silami opačného smeru, ktoré súčasne vznikajú a súčasne zanikajú. Veľkosti párových síl sa rovnajú, ale majú opačné znamienka ($f_i = -tf_j$). To znamená, že diferenciál v Navier-Stokesovej rovnici, ktorý vytvára tieto sily musí byť symetrizovaný.

Monaghan vyvinul symetrizáciu, ktorú nazývame *Symmetric Gradient Approximation Formula* (SGAF). Táto formulácia zahrňuje lineárny a uhlový moment, ale nie tak presne ako DGAF. Bežne sa používa pre tlakový gradient.

$$\nabla A_i = \rho_i \sum_j m_j \left(\frac{A_i}{\rho_i^2} + \frac{A_j}{\rho_j^2}\right) \nabla W\left(\mathbf{r}_{ij}, h\right) \tag{2.21}$$

Alternatívna formulácia symetrie:

$$\nabla A_i = \sum_j \frac{m_j}{\rho_j} \left(A_i + A_j\right) \nabla W\left(\mathbf{r}_{ij}, h\right) \tag{2.22}$$

Kompletná derivácia, formulácie vektora gradientu, vektorový a skalárny súčin, atď. sú vysvetlené v [11].

Korekcia laplaciánu

Pre základné formulácie laplaciánu bolo zistené, že je trochu nestabilná za určitých podmienok, ale existuje široká škála možných opráv. V [28] je popísaná vhodná oprava laplaciánu pre viskóznu silu:

$$\nabla \cdot \left(\frac{1}{\rho} \nabla A\right) = \sum m_j \frac{8}{\left(\rho_i + \rho_j\right)^2} \frac{\left(A_i - A_j\right) \cdot \nabla W\left(\mathbf{r}_{ij}, h\right)}{\left|\mathbf{r}_{ij}\right|^2 + \eta^2} \tag{2.23}$$

kde η je malé číslo a používa sa z dôvodu, aby menovateľ nebol nulový. Zvyčajne sa rovná $0.1 \cdot h$, kde h je dĺžka vyhladzovacieho jadra.

Podrobnejší prehľad o SPH nájdete v [11] a [12].

2.6 Dynamika tekutín

Rovnice 2.3 a 2.4 popisujú základnú eulerovu formuláciu nestlačitelných tekutín. Používanie častíc namiesto mriežky výrazne zjednodušuje rovnice. Predpokladáme, že množstvo častíc je konštantná v priebehu simulácie, udržiavame vopred stanovenú hmotnosť pre každú časticu, to znamená, že zachovanie hmoty je zaručené a rovnicu 2.4 môžeme vynechať.

Obrázok 2.3 znázorňuje základné usporiadanie časticovej tekutiny, ktorá bola redukovaná na 2D z dôvodu prehľadnosti.

V Lagrangeovej formulácii častice definujú tekutinu presne, čo znamená, že sa pohybujú spolu s tekutinou. V porovnaní s Eulerovskou formuláciou to znamená, že všetky parametre závisia len na čase t. Častica nesie zo sebou hmotu, pozíciu a rýchlosť, a bude udržiavať vyhladené aproximácie získané z SPH. Zrýchlenie častice Lagrangeovej tekutiny je potom bežná derivácia rýchlosti $v(t)$ podľa času $\frac{\mathrm{d}}{\mathrm{d}t}$. Toto objasní prečo chýba časť $\left(\frac{\partial v}{\partial t} + v.\nabla v\right)$ (rovnica 2.3) z Lagrangeovej formulácie.

Základná Lagrangeova formulácia rovnice Navier-Stokesa pre nestlačitelnú tekutinu je daná vzťahom:

$$\rho\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = -\nabla p + \mu\nabla^2\mathbf{v} + \mathbf{f} \tag{2.24}$$

kde pravá strana obsahuje interné a externé sily. Tieto sily môžeme sčítať a dostaneme silu $\mathbf{F}$, kde $\mathbf{F} = \mathbf{f}^{internal} + \mathbf{f}^{external}$. Pre i-tú časticu vypočítame zrýchlenie nasledovne:

$$\mathbf{a}_i = \frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = \frac{\mathbf{F}_i}{\rho_i} \tag{2.25}$$

kde $\mathbf{a}_i$ a $\mathbf{v}_i$ sú zrýchlenie a rýchlosť, $\mathbf{F}_i$ je celková pôsobiaca sila a ρ_i je vypočítaná hustota v pozícii i-tej častice.

V časti 2.5 sme sa dozvedeli o prvom zlatom pravidle SPH, ale došli sme k záveru, že izotropné gaussovské jadro nie je vhodné na použitie pre naše účely. Potrebujeme štandardné vyhladzovacie jadro, ktoré má nosiča[1] (angl. compact support) pre medzi-časticové SPH výpočty k riešeniu rovnice 2.24.

V [15] sú popísané rôzne typy jadier na SPH, ktoré majú nosičov funkcií. Medzi tieto jadra patria B-Spline a Q-Spline jadra, kde Q-Spline je považované za najlepšie jadro z hľadiska výpočtovej presnosti.

Avšak Q-Spline jadro vyžaduje vypočítanie druhej odmocniny, ktoré môže byť drahé, časovo náročné, ak jadro je často používané. Namiesto toho budeme používať jadro polynómu 6. stupňa, ktoré navrhli v [22]:

$$W_{poly6}\left(\mathbf{r}, h\right) = \frac{315}{64\pi h^9}\begin{cases}\left(h^2 - |\mathbf{r}|^2\right)^3 & 0 \leq |\mathbf{r}| \leq h \\ 0 & |\mathbf{r}| > h,\end{cases} \tag{2.26}$$

s gradientom:

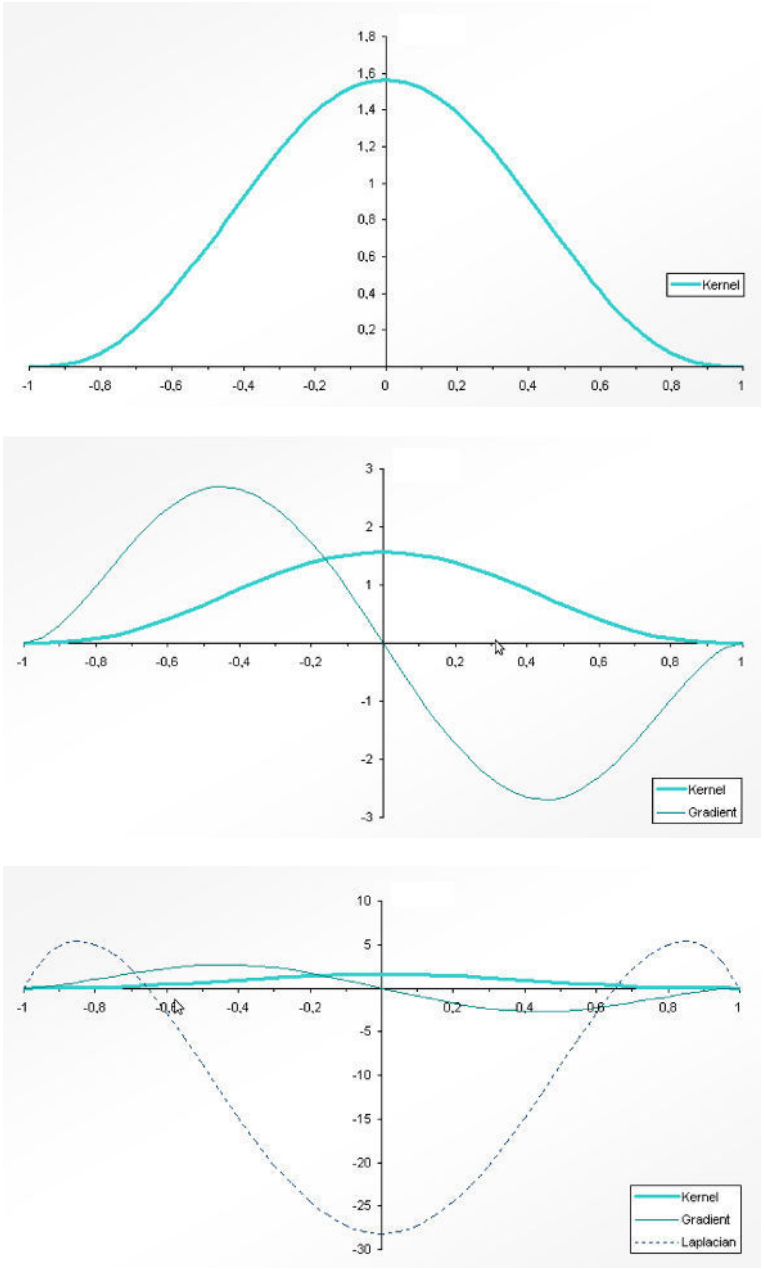
$$\nabla W_{poly6}\left(\mathbf{r}, h\right) = -\frac{945}{32\pi h^9}\mathbf{r}\left(h^2 - |\mathbf{r}|^2\right)^2, \tag{2.27}$$

a s laplaciánom:

$$\nabla^2 W_{poly6}\left(\mathbf{r}, h\right) = -\frac{945}{32\pi h^9}\left(h^2 - |\mathbf{r}|^2\right)\left(3h^2 - 7\,|\mathbf{r}|^2\right). \tag{2.28}$$

Medzi dobré vlastnosti jadra W_{poly6} patria zachovanie zvončekového tvaru gaussovej krivky a to, že nepotrebuje dodatočnú normalizáciu pre $\mathbf{r}$, čo je veľmi výhodné. Obrázok 2.7 znázorňuje jadro W_{poly6} a jeho deriváty v 1D.

[1] Nosičom reálnej funkcie f je podmnožina jej definičného oboru, ktorá obsahuje všetky body x také, že $f(x) \neq 0$.



Obrázek 2.7: Jadro W_{poly6} a jeho derivácie z [21].

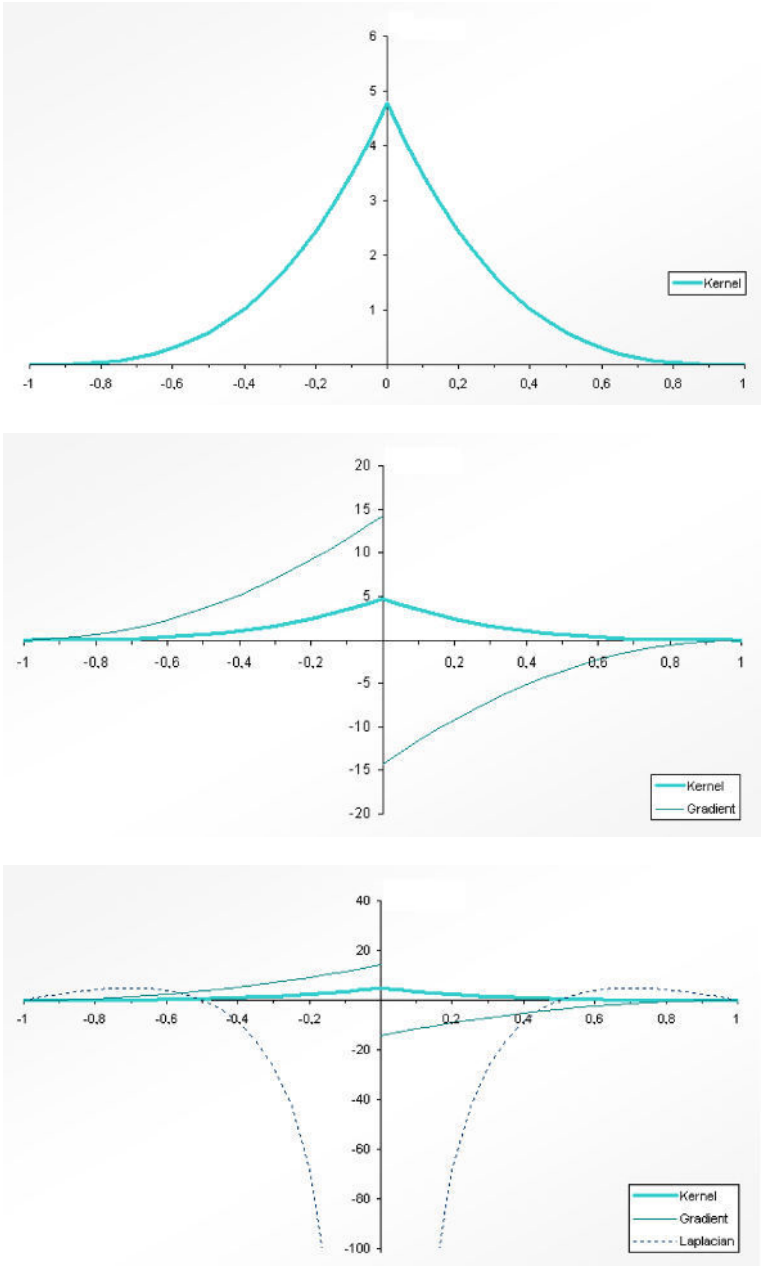
Hustota

Vieme, že všetky vlastnosti, ich gradienty a laplaciány môžeme aproximovať pomocou SPH. Rovnice SPH predpokladajú, že určité vlastnosti sú známe už pred výpočtom (hmota a hustota častice). Hmota častice je užívateľom definovaná konštanta, ale hustota už je spojitá veličina, ktorú musíme vypočítať. Aby sme vylúčili rozpory, hustota je závislá jedine na

hmote častice. Z toho dostaneme:

$$\begin{aligned} \rho_i &= \rho(\mathbf{r}_i) \\ &= \sum_j \rho_j \frac{m_j}{\rho_j} W(\mathbf{r}_i - \mathbf{r}_j, h) \\ &= \sum_j m_j W(\mathbf{r}_i - \mathbf{r}_j, h). \end{aligned} \tag{2.29}$$

Tlak



Obrázek 2.8: Jadro W_{spiky} a jeho derivácie z [21].

Tlak p častice môžeme určiť podľa zákona ideálneho plynu:

$$pV = nRT, \tag{2.30}$$

kde $V = \frac{1}{\rho}$ je objem na jednotku hmoty, n počet plynových častíc na mol, R univerzálna plynová konštanta a T je teplota. Pre izotermické tekutiny s konštantnou hmotnosťou pravá strana rovnice 2.30 sa nemení, a preto môžeme ju nahradiť konštantou plynovej tuhosti k, ktorá teoreticky závisí len od množstva častíc v tekutine. Na základe toho môžeme prepísať rovnicu do tvaru:

$$\begin{aligned} pV &= k \\ p\frac{1}{\rho} &= k \\ p &= k\rho \end{aligned} \tag{2.31}$$

Ak poznáme tlak u každej častice, tlaková sila u i-tej častice u SPH je:

$$\begin{aligned} f_i^{pressure} &= -\frac{1}{\rho}\nabla p\left(\mathbf{r}_i\right) \\ &= -\frac{1}{\rho}\sum_{j\neq i} p_j \frac{m_j}{\rho_j}\nabla W\left(\mathbf{r}_i - \mathbf{r}_j, h\right). \end{aligned} \tag{2.32}$$

Žiaľ tlaková sila v 2.32 nie je symetrická. To je možné overiť napr. vtedy, keď sú len dve častice v interakcii. Prvá častica používa len tlak druhej častice a na základe toho počíta svoju tlakovú silu, a naopak. Pretože tlaky u častíc nie sú rovnaké, tlakové sily budú všeobecne asymetrické a tým bude porušený Newtonov tretí zákon akcie-reakcie. SPH určuje spôsob, ako symetrizovať tlakovú silu:

$$f_i^{pressure} = -\sum_{j\neq i}\left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) m_j \nabla W\left(\mathbf{r}_i - \mathbf{r}_j, h\right). \tag{2.33}$$

Presnejší popis o jednotlivých krokoch dosadenia do rovnice 2.32 je v [21].

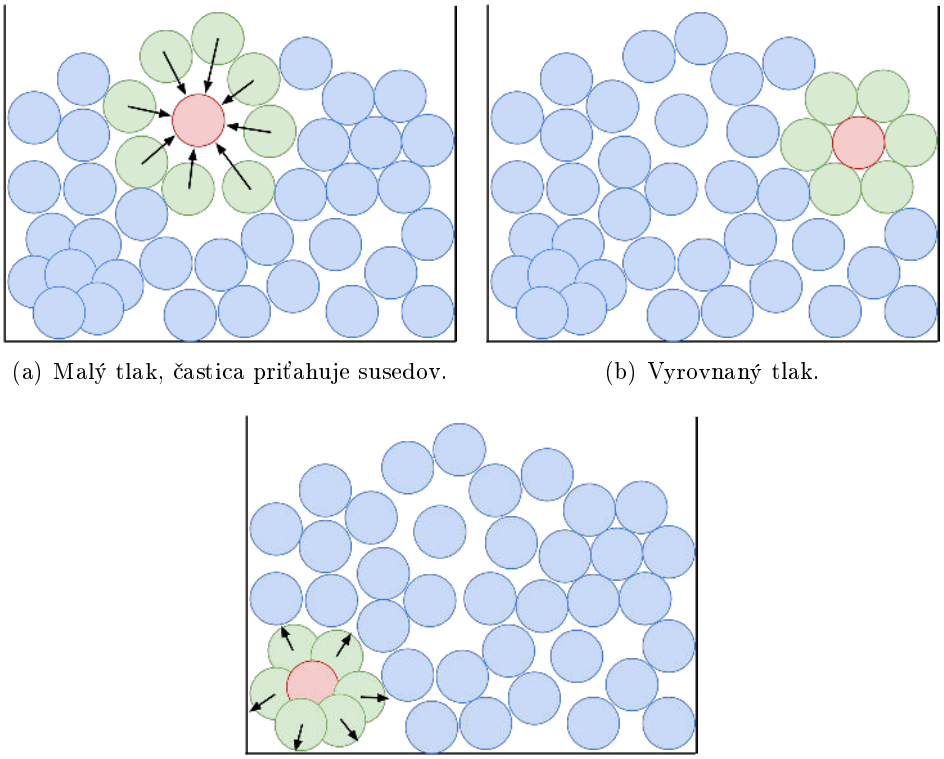
Existuje aj iná možnosť symetrizácie asymetrických síl. V [22] je popísané jednoduché riešenie, ktoré poskytuje dobrú rýchlosť a stabilitu:

$$f_i^{pressure} = -\frac{1}{\rho_i}\sum_{j\neq i}\frac{p_i + p_j}{2}\frac{m_j}{\rho_j}\nabla W\left(\mathbf{r}_i - \mathbf{r}_j, h\right). \tag{2.34}$$

Táto rovnica už produkuje symetrické sily vďaka tomu, že používa štandardný aritmetický priemer interagujúcich častíc. Symetrická tlaková sila zachováva lineárne a uhlové momenty, a teda aj Newtonov tretí zákon.

Výpočet tlaku pomocou 2.34 a jeho použitie pri výpočte tlakovej sily, produkuje vo výsledku čistú silu, ktorá bude odpudzovať častice jednu od druhej. Taká sila pôsobuje v ideálnych plynoch a expanduje častice do priestoru. Napriek tomu kvapaliny vykazujú vnútornú súdržnosť a konštantnú hustotu hmoty v pokoji. V [20] odporúčajú používať upravenú verziu rovnice ideálneho plynu:

$$\begin{aligned} \left(p + p_0\right)V &= k \\ p + k\rho_0 &= k\rho \\ p &= k\left(\rho - \rho_0\right), \end{aligned} \tag{2.35}$$



(a) Malý tlak, častica priťahuje susedov.

(b) Vyrovnaný tlak.

(c) Veľký tlak, častica odpudzuje susedov.

Obrázek 2.9: Znázornenie tlakových síl, pre rôzne hustoty.

kde ρ_0 je kľudová hustota tekutiny. Dosadením rovnice 2.35, ako výpočet aktuálneho tlaku do rovnice 2.33 dostaneme silu, ktorá bude priťahovať i odpudzovať častice. Táto sila sa minimalizuje postupne, čím bude rozdiel menší medzi aktuálnou tlakovou silou a kľudovou silou. Ak je tlaková sila väčšia, bude odpudzovať, ak je menšia, bude priťahovať. Obrázok 2.9 ukazuje správanie častíc v závislosti od rôznych tlakových síl.

Gradient z tlakového poľa sa používa pri výpočte tlakovej sily. Ak používame gradient jadra W_{poly6} v rovnici 2.33, tak častice budú sa zhlukovať v regiónoch s vysokým tlakom. Častice budú vytvárať zhluky kvôli $W_{poly6}(\mathbf{r}, h) \to 0$ ak $|\mathbf{r}| \to 0$ (obr. 2.7). Odpudivé sily budú tým menšie, čím sú častice bližšie k sebe. To znamená, že toto jadro nepopisuje presne fyzikálne vlastnosti tlakovej sily, preto potrebujeme iné. V [20] je popísaný tento problém a je navrhnuté ďalšie normalizované jadro. K popísaniu tlakovej sily pouzívame tzv. *spiky* jadro (špicaté):

$$W_{spiky}(\mathbf{r}, h) = \frac{15}{\pi h^6} \begin{cases} (h - |\mathbf{r}|)^3 & 0 \leq |\mathbf{r}| \leq h \\ 0 & |\mathbf{r}| > h, \end{cases} \tag{2.36}$$

ktorý má gradient:

$$\nabla W_{spiky}(\mathbf{r}, h) = -\frac{45}{\pi h^6} \frac{\mathbf{r}}{|\mathbf{r}|} (h - |\mathbf{r}|)^2, \tag{2.37}$$

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

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

a laplacián:

$$\nabla^2 W_{spiky}(\mathbf{r}, h) = -\frac{90}{\pi h^6}\frac{1}{|\mathbf{r}|}(h - |\mathbf{r}|)(h - 2|\mathbf{r}|), \tag{2.38}$$
$$\lim_{x \to 0^+} \nabla^2 W_{spiky}(r, h) = -\infty,$$

kde limity platia pre 1D. Obrázok 2.8 ukazuje jadro pre tlakovú silu a jeho deriváty. Toto jadro už bude presne modelovať požadovanú odpudzovaciu silu, ak susedné častice budú mať vysokú hustotu.

Viskozita

Tekutina je látka, ktorá neodoláva šmykovému napätiu, preto sa evidentne deformuje. V čase prietoku tekutiny medzi molekuly vzniká vnútorné trenie, čo pôsobuje zníženie kinetickej energie. Táto energia sa premieňa na teplo. Odolnosť proti toku sa nazýva viskozita a koefiecient viskozity μ definuje mieru viskozity danej tekutiny.

V SPH viskozita je daná vzťahom:

$$\begin{aligned} \mathbf{f}_i^{viscosity} &= \mu \nabla^2 \mathbf{v}(\mathbf{r}_i) \\ &= \mu \sum_{j \neq i} \mathbf{v}_j \frac{m_j}{\rho_j} \nabla^2 W(\mathbf{r}_i - \mathbf{r}_j, h). \end{aligned} \tag{2.39}$$

Rovnako ako tlaková sila, viskózna sila je tiež asymetrická kvôli rozdielom rýchlosti medzi časticami. V [22] zvolili nasledujúcu rovnicu, aby symetrizovali rýchlostné pole:

$$\mathbf{f}_i^{viscosity} = \mu \sum_{j \neq i} (\mathbf{v}_j - \mathbf{v}_i) \frac{m_j}{\rho_j} \nabla^2 W(\mathbf{r}_i - \mathbf{r}_j, h), \tag{2.40}$$

čo je možné vďaka tomu, že viskózne sily závisia len od rozdielov medzi rýchlosťami a nie od absolútnych rýchlostí. Ďalej môžeme použiť druhé zlaté pravidlo SPH a umiestniť časť viskozity a hustoty vo vnútri operátora laplaciánu:

$$\mu \nabla^2 \mathbf{v} = \frac{\mu}{\rho}\left(\nabla^2(\rho \mathbf{v}) - \mathbf{v}\nabla^2 \rho\right), \tag{2.41}$$

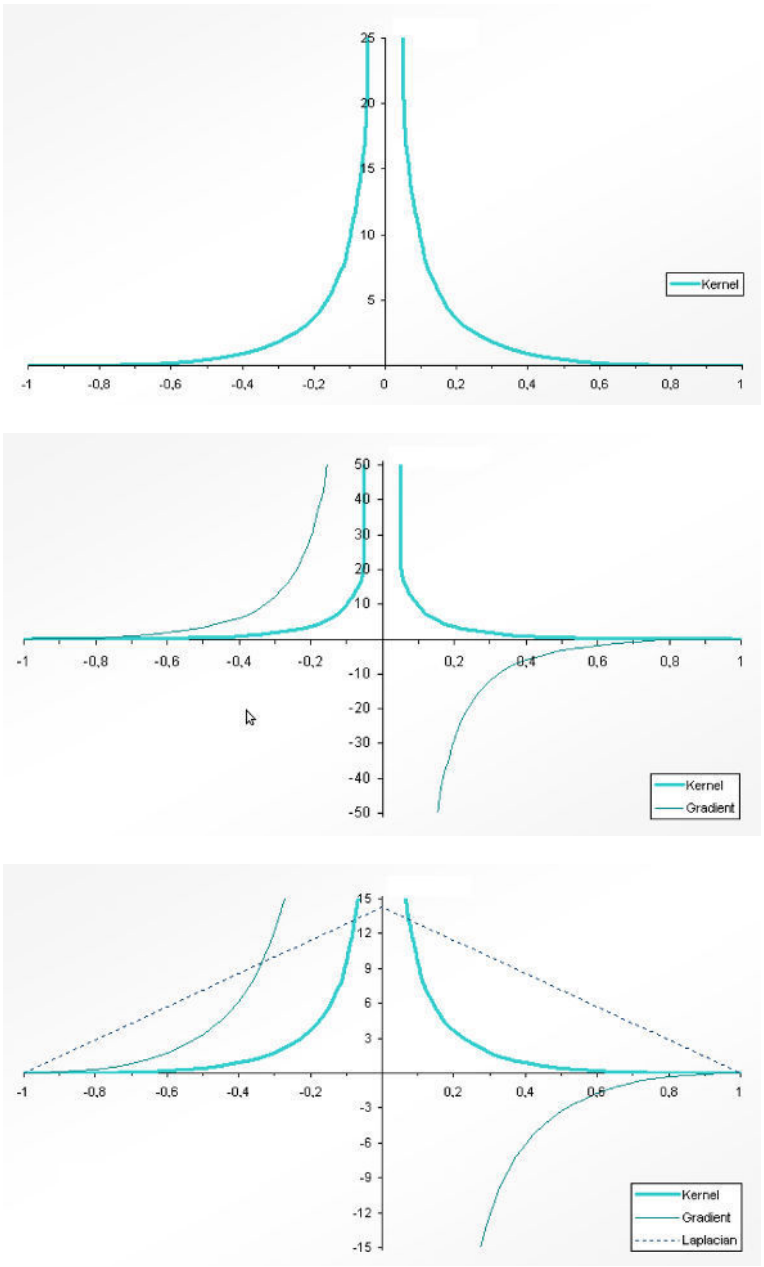
ktorá v SPH bude vyzerať takto:

$$\mathbf{f}_i^{viscosity} = \frac{\mu}{\rho_i} \sum_j (\mathbf{v}_j - \mathbf{v}_i) m_j \nabla^2 W(\mathbf{r}_i - \mathbf{r}_j, h). \tag{2.42}$$

Rovnicu 2.40 môžeme zapísať ako 2.42 za predpokladu, že hustota je konštantná a rovnaká u všetkých častíc. Hustota sa počíta použitím rovnice 2.29, teda všeobecne platí, že sa líši od častice k častici, vidíme, že je lepšie používať rovnicu 2.40 ako 2.42.

Laplacián vyhladzovacieho jadra je obmedzený tak, aby bol pozitívny. Tento krok je nutný, pretože nechceme, aby viskózna sila zvýšila relatívnu rýchlosť častíc a tým by pridala energiu, čo by mohlo spôsobiť nestabilitu systému. Keď je laplacián všade pozitívny, len vtedy bude viskózna sila pracovať ako tlmenie, a bude znižovať relatívnu rýchlosť. Základné jadro W_{poly6} nemá túto vlastnosť a ani jadro W_{spiky}, čo je vidieť i na obrázkoch 2.7 a 2.8. Jadro pre viskozitu budeme definovať podľa [22]:

$$\begin{aligned} W_{viscosity}(\mathbf{r}, h) &= \frac{15}{2\pi h^3} \begin{cases} -\frac{|\mathbf{r}|^3}{2h^3} + \frac{|\mathbf{r}|^2}{h^2} + \frac{h}{2|\mathbf{r}|} - 1 & 0 < |\mathbf{r}| \leq h \\ & |\mathbf{r}| > h, \end{cases} \\ \lim_{x \to 0} W_{viscosity}(r, h) &= -\infty, \end{aligned} \tag{2.43}$$



Obrázek 2.10: Jadro $W_{viscosity}$ a jeho derivácie z [21].

a gradient je:

$$\begin{aligned} \nabla W_{viscosity}(\mathbf{r},h) &= \frac{15}{2\pi h^3}\mathbf{r}\left(-\frac{3|\mathbf{r}|}{2h^3}+\frac{2}{h^2}-\frac{h}{2|\mathbf{r}|^3}\right) \\ \lim_{x\to 0^-}\nabla W_{viscosity}(r,h) &= +\infty, \\ \lim_{x\to 0^+}\nabla W_{viscosity}(r,h) &= -\infty, \end{aligned} \tag{2.44}$$

a laplacián:

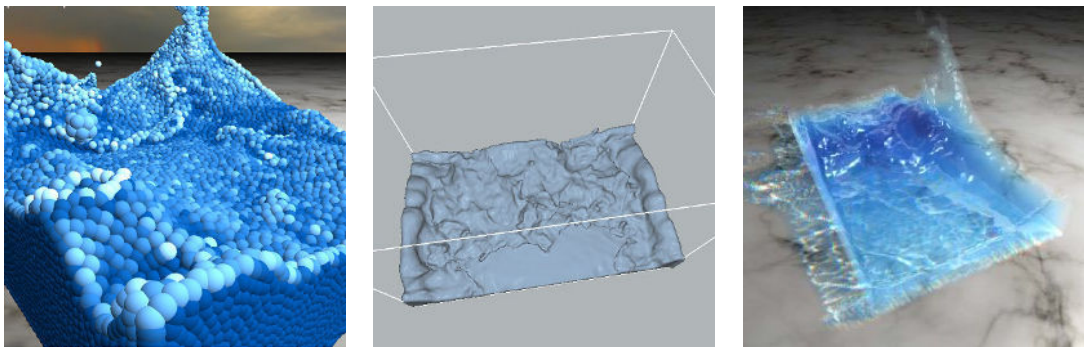
$$\nabla^2 W_{viscosity}(\mathbf{r},h) = \frac{45}{\pi h^6}(h-|\mathbf{r}|), \tag{2.45}$$

kde limity platia pre 1D. Obrázok 2.10 ukazuje jadro viskozity a jeho derivácie.

Kapitola 3

Zobrazenie kvapalín

Dôležitou súčasťou každej simulácie je kvalitné zobrazenie výsledkov. Bez vizuálneho zobrazenia by sme väčšinou nedokázali interpretovať výstup simulácie. U častícových systémoch je to skoro tak dôležité, ako samotná simulácia. Výber vhodnej metódy zobrazenia závisí na konkrétnom type simulácie. Ak chceme kvalitnejšie, realistickejšie výsledky a nezáleží nám na dobe trvania simulácie, môžeme vybrať nejaké *offline* zobrazenia. Offline zobrazenia väčšinou generujú postupne snímky, ktoré sú pospájané do videa. Taká metóda je napríklad *ray-tracing*. Veľkou nevýhodou offline metód je, že nie je možná interakcia medzi simuláciou a užívateľom. Pre naše účely potrebujeme metódu, ktorá dokáže pracovať v reálnom čase. Tieto metódy sa nazývajú *online* metódy.



(a) Point sprites. (b) Marching cubes. (c) Ray Casting.

Obrázek 3.1: Online metódy zobrazenia.

Najpoužívanejšie metódy sú nasledujúce:

- Point Sprites,
- Marching cubes,
- Ray-casting.

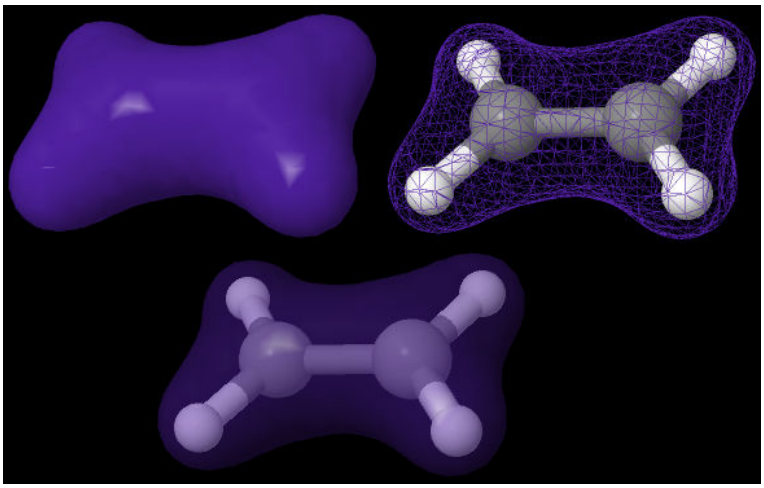
Point sprites je najjednodušia a najrýchlejšia metóda. Používa sa najčastejšie kvôli jednoduchosti a rýchlosti. Okolo častice je vytvorený kruh, do ktorého pridáme tienenie, aby to vyzeralo ako guličkа v 3D. Táto metóda je veľmi dobrá k tomu, aby sme presne videli ako sa chovajú jednotlivé častice. Ostatné metódy generujú objemové telesá, u ktorých už nie je vidieť podpobnosti jednoznačne.

Ďalsia metóda je Marching cubes. Táto metóda generuje trojuholníky, ktoré vytvoria polygonálnu sieť (hraničnú reprezentáciu) a tým dostaneme súvislé plochy. Generované plochy budú sa viac podobať na povrch tekutiny.

Posledná metóda funguje na princípe vzorkovania. Tá už funguje pomalšie ako predchádzajúce metódy. Základ tejto metódy je intersekcia bunky s vrhaným lúčom, teda potrebujeme mriežku na rozdelenie častíc do buniek. Za cenu času dokáže generovať oveľa kvalitnejšie a realistickejšie výsledky, ale potrebuje značné optimalizácie a pobeží v reálnom čase len na najmoderných a najsilnejších kartách.

Častice, ktoré používame v simulácii môžeme chápať, ako nekonečne malé body. Každá častica má svoje okolie definované polomerom. Tvar, ktorý popisujeme stredovým bodom a polomerom v trojrozmernom prostredí je guľa. Pospájaním častíc dostaneme izoplochu, ktorá popisuje povrch simulovaného materiálu.

Izoplocha

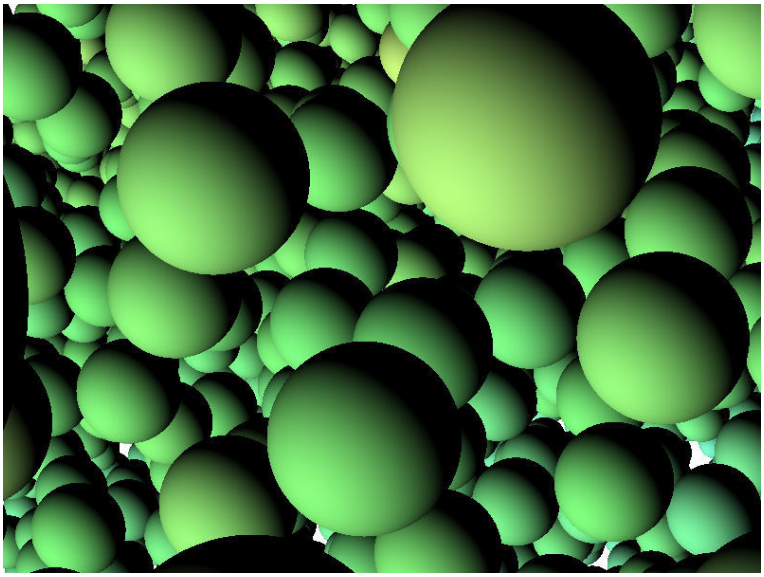


Obrázek 3.2: Izoplocha (molekula eténu) z [3].

Čo je to izoplocha? Najjednoduchšie sa dá vysvetliť na úrovni atómov. Máme jadro atómu a okolo neho sa pohybujú elektróny. Pozíciu týchto elektrónov nie možné určiť presne v danom okamihu, ale sme schopní určiť 99% - nou pravdepodobnosťou ich výskyt v nejakej vzdialenosti od jadra. Takto dostaneme orbitály eletrónov, ktoré popisujú volumetrické dáta. Vzdialenosťou eletrónov od jadra klesá hustota orbitálu, teda pravdepodobnosť ich výskytu. Z týchto dát chceme vytvoriť izoplohu, preto zvolíme nejakú hustotu (prah). V oblastiach kde hustota klesá pod prah alebo stúpa nad prah, budeme definovať izoplochu. Na obrázku 3.3 je vidieť molekulu eténu a izoplochu, ktorú generujú jej atómy pre určitú hustotu.

3.1 Point sprites

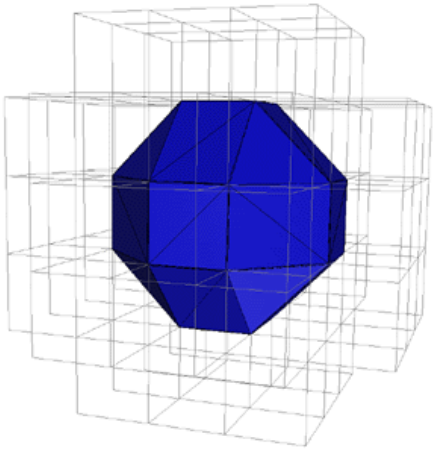
Táto metóda predstavuje najjednodušie možné riešenie na zobrazenie častíc. Každá častica je zobrazená ako gulička. Algoritmus sa implementuje v shaderoch grafickej jednotky. Pri behu sa používa vertex a frament shader. Vo vertex shaderi, v bode každej častice generujeme štvorce. Veľkosť štvorca je závislá na vzdialenosti medzi časticou a kamerou. Fragment shader je zopovedný za zahodenie pixelov mimo polomer guličiek.. Po zahodení dostamene



Obrázek 3.3: Point sprites z blízka.

kruh, ktorý musíme ešte tieniť. Ak chceme, aby bolo vyriešené aj čiastočné prekrývanie guličiek je nutné upraviť Z-buffer pre každý pixel, lebo všetky pixely guličky sú na rovnakej hĺbke.

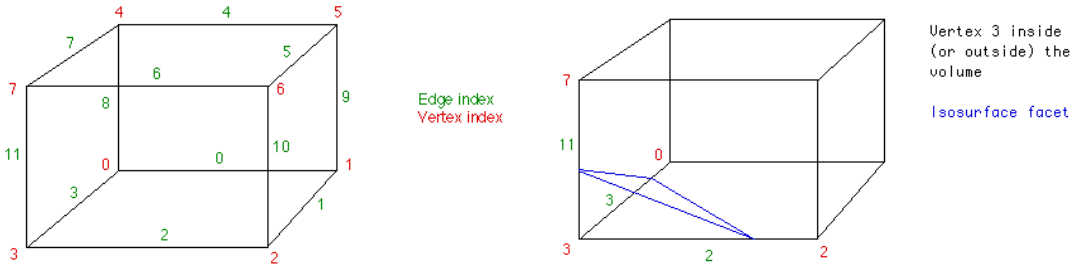
3.2 Marching cubes



Obrázek 3.4: Marching cubes, imaginárne kocky, prevzatý z [24].

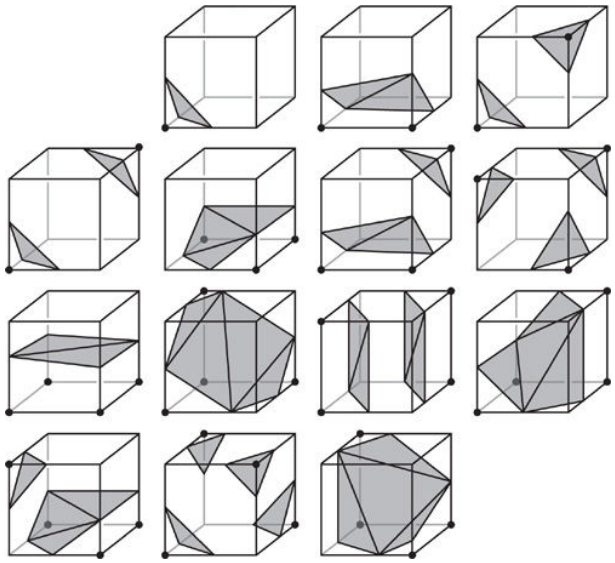
Marching cubes je všeobecná metóda, ktorá sa používa na extrahovanie trojuholníkovej siete (izoplochy) z volumetrických dát. Vzhľadom k tomu, že ide o relatívne starú metódu, bola publikovaná v roku 1987, ešte stále sa používa. Algoritmus prechádza cez skalárne pole, pričom zoberie v danom mieste vždy osem susedov. Susedia tvoria imaginárne kocky, ktoré sú na obrázku 3.4. Na aktuálnu kocku určí potrebný počet trojuholníkov na základe vopred vypočítaných konfigurácií. (obr. 3.6. Maximálný počet konfigurácií je 2^8, lebo kocka má 8

vcholov a každý vrchol má svoju skalárnu hodnotu, ktorá je buď vyššia alebo nižšia ako iso-hodnota (prah). V našom prípade tá skalárna hodnota bude hustota vypočítaná podľa počtu častíc v danom okolí. Vrcholy a hrany sú očíslované, každý má svoj index (obr. 3.5). Tieto indexy slúžia k vytvoreniu osembitového pola. Ak je vrchol nad iso-hodnotou, nastavíme bit



Obrázek 3.5: Očíslované vrcholy a hrany v Marching cubes z [24].

na jednotku, inak bude nula. Tento osembitový vektor bude ukazovať do look-up tabuľky, ktorá má predpočítané pretínané hrany a počet potrebných trojuholníkov k pokrytiu plochy. Tabuľka je dostupná v [24]. Podľa pretínaných hrán môžeme jednoducho vypočítať vrcholy trojuholníkov. Na obrázu 3.5 na pravej strane vidíme, že vrchol s indexom 3 je pod iso-hodnotou, preto vrcholy trojuholníkov budú umiestnené na hranách 11, 3 a 2. Ak chceme



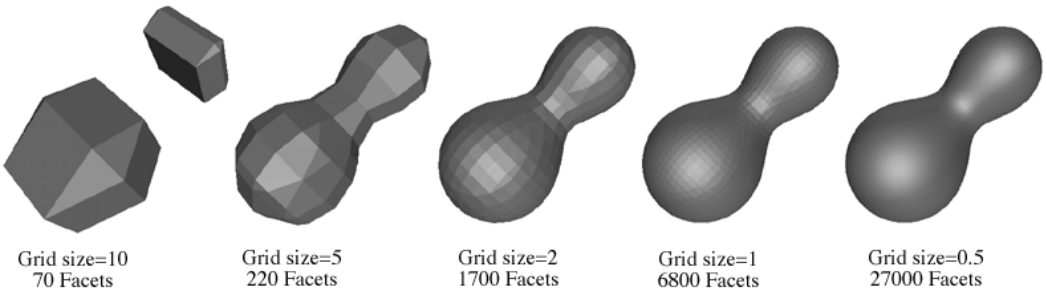
Obrázek 3.6: Konfigurácia trojuholníkov podľa vrcholov [27].

dostať krajšie výsledky, musíme pridať do výpočtu vrcholov trojuholníkov aj skalárne hodnoty vo vrcholoch imaginárnej kocky. Tieto hodnoty použijeme k váhovaniu pozícií vrcholov na pretínanej hrane medzi vrcholmi kocky. Vzorec na získanie pozícií vrcholov trojuholníkov je:

$$P = P_1 + (isoValue - V_1)\,(P_2 - P_1)\,/\,(V_2 - V_1)\,, \tag{3.1}$$

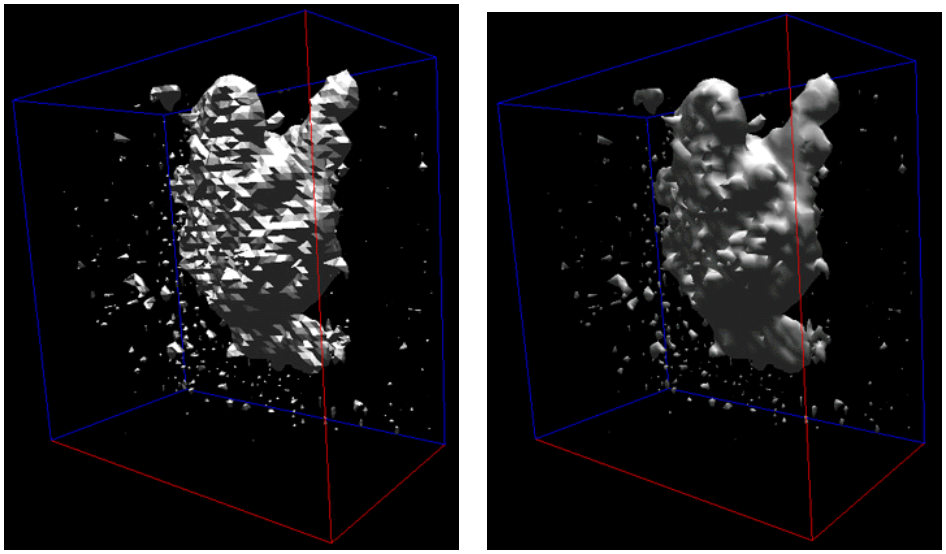
kde P je výsledná pozícia na hrane, P_1 a P_2 je začiatok a koniec hrany, V_1 je iso-hodnota v bode P_1 a V_2 je iso-hodnota v bode P_2.

Algoritmus má niekoľko pozitívnych vlastností kvôli ktorým sa používa aj u časticových systémoch. Ide o algoritmus, ktorý je dobre paralelizovateľný, teda umožňuje využívať GPU na generovanie trojuholníkov. Navyše sa dá používať mriežka simulácie, do ktorej sú umiestnené častice, čím sa ušetrí veľa práce. Napriek dobrých vlastností má aj svoje nedostatky



Obrázek 3.7: Závislosť hladkosti povrchu na rozlíšení mriežky v metóde Marching cubes z [24].

a obmedzenia. Ak si zvolíme veľmi malé imaginárne kocky pri generovaní trojuholníkov, môžeme dostať drsný povrch, a naopak, keď si zvolíme veľké, nebudeme zachytávať kvapky tekutiny. Závislosť výsledkov od rozlíšenia mriežky a veľkosti kociek ukazuje obrázok 3.7. Ďalšia negatívna vlastnosť je, že v dobe generovania trojuholníkov nie je možné generovať normály k vrcholom. K tomu, aby to bolo možné, potrebovali by sme aj vrcholy susedných trojuholníkov, ale kvôli paralelnému výpočtu nie je to možné. Je to samozrejme voliteľný krok, ale značne vylepší finálne výsledky. Porovnanie výsledkov bez a s vypočítanými normálami je vidieť na obrazu 3.8. V prípade, že máme vypočítané korektné normály, sa dá zapnúť hardwarová tessellácia a tým obídeme mriežky s veľkým rozlísením. Vypočítame len hrubý povrch a pomocou tessellácie postupne zjemňujeme. O tessellácii viac v [4] a [2], a o delení plochy v [1] a [9].



Obrázek 3.8: Vľavo bez vypočítaných normál, vpravo s normálmi v metóde Marching cubes z [24].

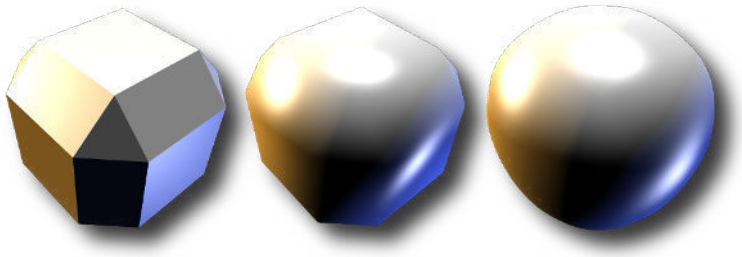
Tessellácia

Tessellácia sa používa k deleniu priamkov, trojuholníkov, štvorcov na menšie časti, tak aby konzistencia modelov bola zachovaná. Pomocou tessellácie je možné odstrániť ostré hrany a vytvoriť hladké povrhy, alebo s využitím deformačných máp vytvoriť z jednoduchších vysoko detailné modely. Nová, 4. verzia OpenGL dostala nové fázy spracovávania geometrie speciálne pre záplaty. Má dve dodatočné programovateľné fáze: control shader a evaluation shader, ktoré slúžia na tesselláciu.

Control shader sa programuje v jazyku GLSL. Cieľom tejto jednotky je určiť úroveň delenia povrchu primitívov. Táto jednotka je zavolaná raz pre každú primitívu počas spracovávania.

Evaluation shader sa začína používať vo fáze, keď tessellátor ukončil delenie záplat. Táto jednotka sa programuje tiež v jazyku GLSL, ktorý je určený pre túto jednotku. V tejto fáze môžeme vypočítať výsledné požície vytvorených nových vrcholov a normálov. Táto jednotka je zavolaná raz pre všetky nové vrcholy, ktoré boli vytvorené vo fáze tessellácie.

Na delenie trojuholníkov a na deformácie povrchu, aby sme ostránili ostré hrany budeme používať metódu zakrivených PN trojuholníkov.



Obrázek 3.9: Hladkosť povrchu v závislosti na úrovne tessellácie z [8].

Zakrivené PN Trojuholníky

Metóda zakrivených PN (point-normal) trojuholníkov bola prvýkrát predstavená v roku 2001 v článku "Curved PN Triangles"[7]. Túto primitívu môžeme vytvoriť z obyčajného trojuholníka (ktorý obsahuje tri vrcholy a normály - PN) so zakombinovaním normálového pola (normal field) a zdvihového pola (displacement field). Túto metódu budeme používať na elimináciu ostrých hrán generovaných metódou Marching cubes.

Zdvihové pole

Nech je trojuholník definovaný troma vrcholmi $V_i = (P_i, N_i)$, kde P_i je pozícia, a N_i je normalizovaný normál i-tého vrcholu. Zdvihové pole $b(u,v)$ PN trojuholníka je definované nasledovne:

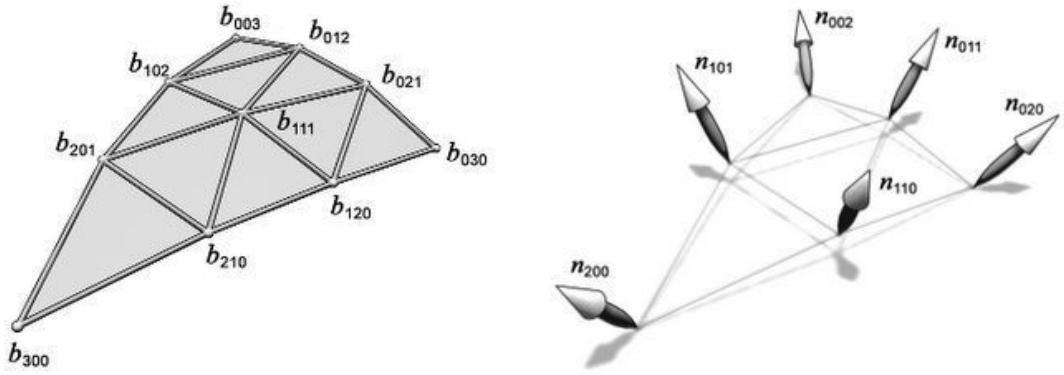
$$\begin{aligned} b(u,v) &= \sum_{i+j+k=3} b_{ijk} \frac{3!}{i!j!k!} u^i v^j w^k \qquad (3.2) \\ &= b_{300}w^3 + b_{030}u^3 + b_{003}v^3 + \\ &\quad b_{210}3u^2v + b_{120}3uv^2 + b_{201}3u^2w + \\ &\quad b_{021}3v^2w + b_{102}3uw^2 + b_{012}3vw^2 + \\ &\quad b_{111}6uvw, \end{aligned}$$

kde (u, v, w) sú relatívne barycentrické súradnice od vrcholov a term b_{ijk} značí kontrolné body PN trojuholníka.

Ak definujeme $w_{ij} = (P_j - P_i) \cdot N_i$, potom kontrolné body sú:

$$\begin{aligned}
b_{300} &= P_1, \\
b_{030} &= P_2, \\
b_{003} &= P_3, \\
b_{210} &= \frac{1}{3}\left(2P_1 + P_2 - w_{12}N_1\right), \\
b_{120} &= \frac{1}{3}\left(2P_2 + P_1 - w_{21}N_2\right), \\
b_{021} &= \frac{1}{3}\left(2P_2 + P_3 - w_{23}N_2\right), \\
b_{012} &= \frac{1}{3}\left(2P_3 + P_2 - w_{32}N_3\right), \\
b_{102} &= \frac{1}{3}\left(2P_3 + P_1 - w_{31}N_3\right), \\
b_{201} &= \frac{1}{3}\left(2P_1 + P_3 - w_{13}N_1\right), \\
b_{111} &= E + \frac{1}{2}\left(E - V\right),
\end{aligned}$$

kde $E = \frac{1}{6}\left(b_{210} + b_{120} + b_{021} + b_{012} + b_{102} + b_{201}\right)$ a $V = \frac{1}{3}\left(P_1 + P_2 + P_3\right)$.



(a) Kontrolné body PN trojuholníka. (b) Normály PN trojuholníka.

Obrázek 3.10: Delenie povrchu metódou zakrivených PN trojuholníkov[7].

Normálové pole

Normálové pole môžeme popísať kvadratickou funkciou $n(u, v)$, ktorá je definovaná nasledovne:

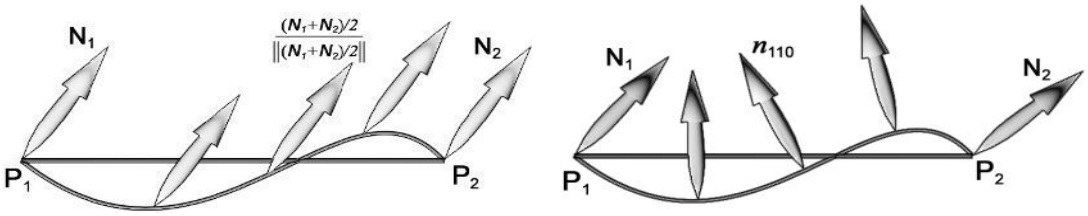
$$n(u, v) = \sum_{i+j+k=2} n_{ijk} u^i v^j w^k \tag{3.3}$$

$$n_{200}w^2 + n_{020}u^2 + n_{002}v^2 + n_{110}uv + n_{011}vw + n_{101}wv \tag{3.4}$$

Definíciou $v_{ij} = 2\frac{(P_j - P_i)\cdot(N_j + N_i)}{(P_j - P_i)\cdot(P_j - P_i)}$ dostaneme:

$$\begin{aligned}
n_{200} &= N_1, \\
n_{020} &= N_2, \\
n_{002} &= N_3, \\
n_{110} &= \frac{N_1 + N_2 - v_{12}\,(P_2 - P_1)}{|N_1 + N_2 - v_{12}\,(P_2 - P_1)|} \\
n_{011} &= \frac{N_2 + N_3 - v_{23}\,(P_3 - P_2)}{|N_2 + N_3 - v_{23}\,(P_3 - P_2)|} \\
n_{101} &= \frac{N_3 + N_1 - v_{31}\,(P_1 - P_3)}{|N_3 + N_1 - v_{31}\,(P_1 - P_3)|}
\end{aligned}$$

Dôvodom požívania kvadratickej funkcie je, že s lineárnou funkciou nie je možné presne vyjadriť zakrivenie plochy. Túto situáciu ukazuje obrázok 3.11.



(a) Lineárna. (b) Kvadratická.

Obrázek 3.11: Závislosť normálového pola na stupne funkcie.

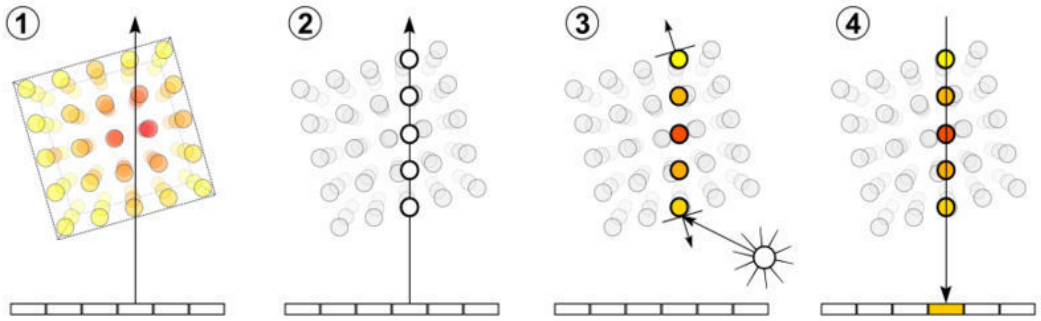
Podobná metóda, ktorá sa používa na odstránenie ostrých hrán sa nazýva Phongova tessellácia (Phong tessellation). Táto metóda bola predstavená v roku 2008, podrobnosti je možné nájsť v [29].

3.3 Volume Ray Casting

Volume ray casting, niekedy volumetric ray casting je technika na rendrovanie trojrozmerných objemových dát na dvojrozmerné plátno. Metóda Volumetric ray casting, ktorá spracováva objemové dáta, je často pomýlená metódou ray casting, ktorá spracováva povrchové dáta. Táto metóda je najlepšia voľba k dosiahnutiu najrealistickejších výsledkov v porovnaní metódou Marching cubes a Point Sprites. Metóda Ray casting podporuje výpočet priesvitnosti, reflekcie a refrakcie. Tieto tri javy sú veľmi dôležité u kvapalín. Samozrejme výpočet realistických javov má svoju cenu, výpočty potrebujú viac času.

Základný algoritmus má 4 hlavné kroky:

- **Ray casting**. Pre každý pixel výsledného obrazu je strielaný pohladový lúč cez objem. V tejto fázi je dobré používať obalové teleso, jednoduché geometrické teleso, ktoré sa používa na rýchlu detekciu pretínania lúčom.
- **Vzorkovanie**. Podĺž smeru lúča vyberáme vzorky v konštantnej vzdialenosti.
- **Tienenie**. Pre vzorky vypočítame gradient osvetlenia. Tieto hodnoty predstavujú lokálne povrchy v objeme. Vzorky sú potom tienené, farbené v závislosti na povrchu, orientácie a pozície svetla v scéne.

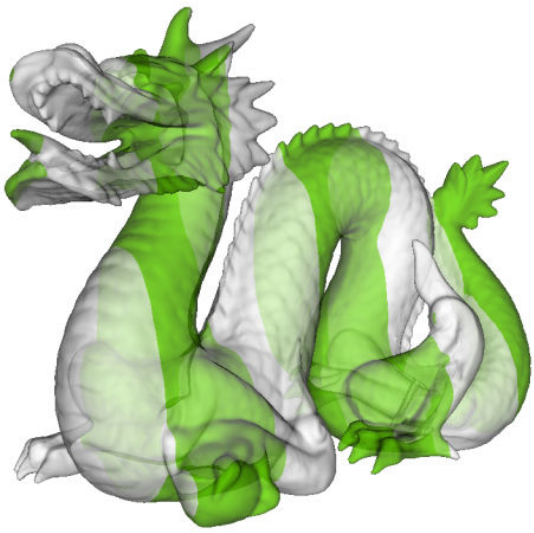


Obrázek 3.12: Štyri hlavné kroky metódy ray casting: (1) Ray casting (2) Vzorkovanie (3) Tienenie (4) Skládanie z [32].

- **Skladanie**. Ak už vzorky boli tienené, môžeme ich skladať, a vypočítať finálnu farbu pre pixel. Pri skladaní postupujeme od zadných po predné vzorky.

Depth peeling

Metódu Ray casting je možné zrýchliť pomocou metódy Depth peeling, ktorá je založená na viacnásobnom renderovaní scény do textúr. V každom kroku generujeme textúru, ktorá bude obsahovať hĺbky hĺbších pixelov ako v predchádzacúcom kroku. Táto jednoduchá metóda umožňuje preskákať prázdne miesta a tým sa výpočet značne urýchli.



Obrázek 3.13: HĺBkové plátky v metóde depth peeling z [19].

Kapitola 4

Návrh

V tejto kapitole popíseme návrh aplikácie. Kapitola obsahuje metódy a techniky, ktoré boli používané počas implementácie. Rozoberieme možnosti paralelných výpočtov na grafickej karte, integráciu síl v metóde SPH, atď.

4.1 Paralelné výpočty

Tento prístup k výpočtom je významný preto, lebo môžeme výpočty vykonávať súbežne. Vzhľadom k tomu, že metóda SPH je dobre paralelizovatelná, je dobrá voľba využívať silu moderných grafických kariet na výpočty časticových simulácií. Existuje niekoľko rôznych foriem tohto princípu, od paralelizmu na úrovni bitov, na úrovni inštrukcií (napr. pipelining v moderných CPU), dátový paralelizmus (rovnaké výpočty na rôzne údaje) a paralelizmus úloh (rôzne výpočty na rovnaké dáta). Tieto rôzne formy boli prvýkrát zaradené M. J. Flynnom, ktorý vytvoril jeho známu taxonómiu:

- SISD - Single Instruction, Single Data stream,
- SIMD - Single Instruction, Multiple Data stream,
- MISD - Multiple Instruction, Single Data stream,
- MIMD - Multiple Instruction, Multiple Data stream.

My sa sústredíme na paralelizmus typu SIMT (Single Instruction, Multiple Threads), je skoro to isté čo SIMD, ale SIMT bol zavedený firmou NVIDIA pre grafické karty. Tento typ je založený na vláknach, ktoré vykonávajú rovnaké výpočty rovnakými dátami tým, že rozložia prácu medzi sebou.

GPGPU

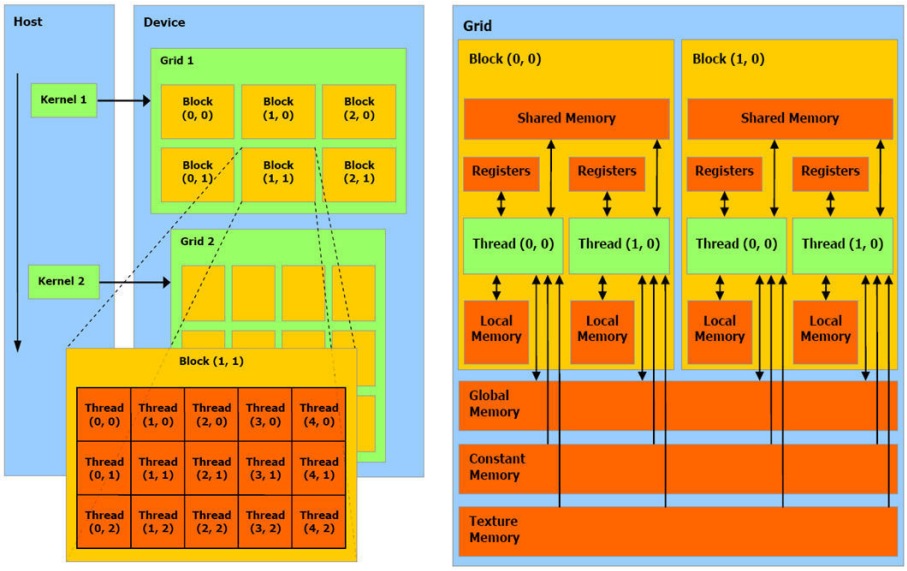
Dnes už asi neexistuje počítač, notebook, či smartphone, ktorý by nemal v sebe GPU. Sme svedkami začínajúcej revolúcie, keď sa snažíme paralelizovatelné výpočty preniesť na GPU. Moderné GPU môže poskytnúť masívny nárast výpočetného výkonu pre dobre paralelizovatelné problémy, ako je napr. dekódovanie videa, rozpoznávanie obrazu, fyzikálne simulácie, atď. GPGPU je skratka *General Purpose Graphics Processing Unit*, čo znamená, že využívame silu grafickej karty na všeobecné, negrafické výpočty. Tento koncept umožňuje simulovať napr. časticové simulácie s veľkým počtom častíc v reálnom čase, ktoré ešte pred niekoľkými rokmi neboli možné. Toto vylepšenie GPU znamená, že vedci sú schopní rýchlo

a ľahko vizualizovať problémy. Týka sa to nielen fyzických problémov, ale aj chemických (syntéza proteínov) a medicínskych (vizualizácia MRI výsledkov v 3D v reálnom čase).

Koncept má samozrejme aj nedostatky, čo je kopírovanie dát z CPU do GPU a naopak. Kvôli priepustnosti zberníc a častým kopírovaním môžeme značne degradovať rýchlosť. Optimálne je vyhnúť sa kopírovaniu, pokiaľ je to možné. Ak chceme výpočty zobraziť napr. pomocou *OpenGL*, netreba prekopírovať z GPU do CPU a potom zase do GPU. Po prvé, duplikovali by sme dáta na GPU a po druhé, zbytočne by sme zdržiavali GPU. Architektúra umožňuje používať vyrovnávaciu pamäť pre výpočty i zobrazenie.

CUDA

CUDA je architektúra grafických kariet vyvinutá firmou NVIDIA. Skrakta znamená *Compute Unified Device Architecture*, čo je zjednotená architektúra pre všeobecné výpočty. Využívať grafickú kartu na všeobecné výpočty už bolo možné aj pred vyvinutím CUDA, ale len v programovateľných *shaderoch*, čo bolo veľmi komplikované a obmedzené. Na uloženie výsledkov používali textúry. Simulácia častíc patrí práve medzi všeobecné výpočty a vďaka tejto platformy je jednoduchšie implementovatelné. Firma NVIDIA vydala kompletné API, ktoré je orientované na výpočty a nie na grafiku, preto je možné oveľa rýchlejšie a jednoduchšie programovať. Podporuje vývojárom používať celé čísla, bitové operácie, zdieľanú pamäť, atď. Ďalšie zaujímavosti sú podrobnejšie popísané v [5].



Obrázek 4.1: Architektúra CUDA, vľavo štruktúra mriežky, vpravo typy pamäte.

Programovací model CUDA vyzerá nasledovne:

- paralelné výpočty bežia v jadrách (kernels)
- možnosť spusiť súbežné, ale rôzne jadrá
- vo vláknach jadier beží rovnaký zdrojový kód paralelne, ale na rôznych kusoch dát.

Základný koncept je taký, že jadro je funkcia, ktorá spúšťa jednotlivé vlákna, kde každé vlákno má svoj index. Na základe indexu je možné rozhodovať, ktoré vlákno bude pracovať

na ktorej časti dát. Vlákna sú organizované do blokov a tieto bloky zase do mriežky, čo je vidieť na ľavej strane obrázku 4.1. Blok i mriežka môze mať určitý počet dimenzií, počet a veľkosť sú limitované hardwarom.

V CUDA existuje niekoľko typov pamätí. Prístup do pamäti je často hlavnou príčinou zpomalenia, preto je dôležité dôkladne zoznámiť sa s rôznymi obmedzeniami a výhodami jednotlivých pamätí. Hlavné typy pamäti v CUDA sú tieto:

- hostiteľská pamäť (CPU),
- globálna pamäť (GPU),
- a on-chip pamäť.

Typy pamätí na GPU

Jednotlivé typy pamätí sú vyznačené na obraze na pravej strane 4.1:

- register,
- lokálna pamäť,
- zdieľaná pamäť,
- konštantná pamäť,
- globálna pamäť,
- pamäť textúr.

Prístup do lokálnej pamäte je rovnako drahý ako, prístup do globálnej pamäte. Ak dôjde k prístupu do lokálnej pamäte, znamená že nie je dostatok registrov k dispozícii. Použitiu tejto pamäte je lepšie sa vyhnúť, ak je to možné.

Konštantná pamäť je časť globálnej pamäte, ktorá je zakešovaná. Prvý prístup do tejto pamäte bude vyvolávať cache-miss a dáta sa načítajú z globálnej pamäte, ale ďalšie prístupy už majú dáta k dispozícii za jeden takt. Túto časť budeme používať na predpočítané dáta, ktoré vopred umiestníme do konštantnej pamäte.

Globálna pamäť je najväčšia dostupná pamäť na grafickej karte. Prístup do globálnej pamäte je drahý, pretože je potrebné dbať na prístup z iných vlákien. Dnešné grafické karty už majú obvykle okolo 1-2GB. Je možné používať kešovanie globálnej pamäte do pamäte textúr, čo obecne umožňuje zlepšiť výkon, hlavne vtedy, keď čítame dáta z jednej lokality. Do globálnej pamäte budú uložené všetky vlastnosti častíc, ako hustota, tlaková sila, pozícia, atď.

Na zdieľanú pamäť, čo je on-chip pamäť, sa môžeme pozerať, ako užívateľom riadený cache. Veľkosť zdieľanej pamäte na dnešných grafických kartách sa pohybuje okolo 64KB, čo pri určitých problémoch môže byť málo. Rýchlosť takej pamäte za dobre organizovaných prístupov je porovnateľná rýchlosťou registrov. Zdieľanú pamäť budeme potrebovať pri výpočtoch, v ktorých na jednej úlohe pracuje viac vlákien súbežne a zdielajú výsledky medzi sebou, napr. nájdenie začiatku a konca buniek uniformnej mriežky, vid. kapitola 5.1.

Každé vlákno má prístup do zdieľanej pamäte v danom bloku, kde môžu vymieňať dáta medzi sebou.

4.2 Návrh implementácie metódy SPH

Teoretické základy, differenciálne rovnice, ich diskretizácia, výpočet tlaku a síl metódy SPH sú popísané v kapitole 2. Na simulovanie kvapaliny budeme používať platformu CUDA. To znamená, že je nutné rozdeliť dátovo závislé výpočty:

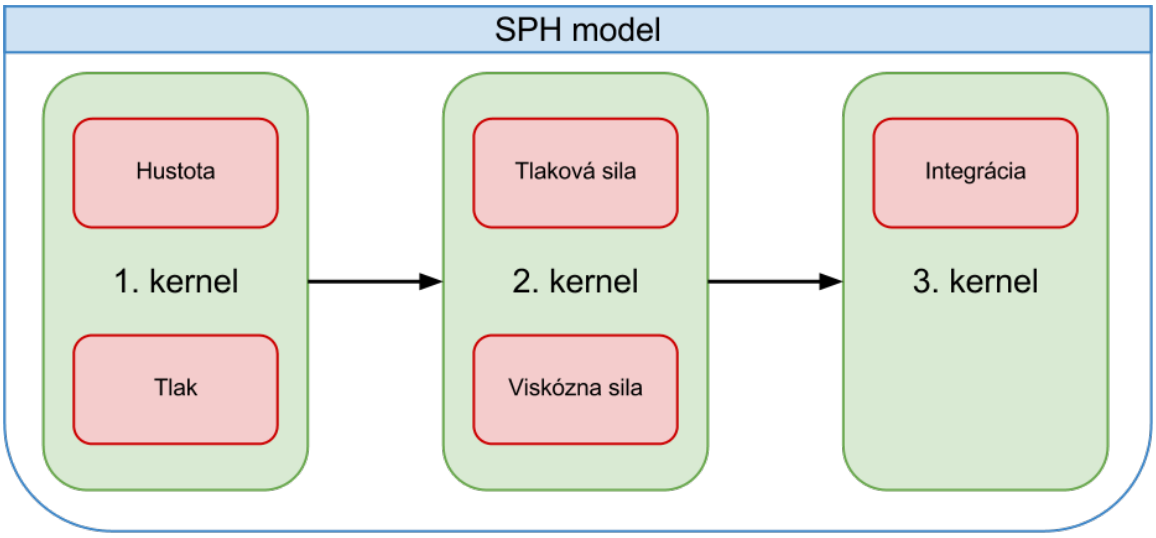
- výpočet hustoty,
- výpočet tlaku,
- výpočet tlakovej a viskóznej sily,
- výpočet rýchlosti,
- výpočet pozície.

Hustota podľa rovnice 2.29 je závislá len od hmoty častice. Hmotu častice poznáme, je to predom nastavená hodnota. K vypočítaniu tlaku v bode častice potrebujeme hustoty okolitých častíc. To znamená, že tieto dva kroky sa dajú vypočítať spoločne v jednom kerneli grafickej karty.

Tlakovú (rovnica 2.34) a viskóznu (rovnica 2.42) silu je možné tiež vypočítať spoločne. Obe sily sú závislé od hustoty, ale tú hodnotu už poznáme. Ďalej, tlaková sila je závislá od tlaku, čo bolo tiež vypočítané v predchádzajúcom kroku. Aby sme vedeli vypočítať viskóznu silu je nutné poznať rýchlosť častíc. Na začiatku simulácie je to buď nula, alebo iná hodnota, na ktorú bola inicializovaná.

Posledný krok je integrácia zrýchlenia. Zrýchlenie dostaneme podľa rovnice 2.25. Vidíme, že všetky hodnoty poznáme, potrebujeme už len integrovať zrýchlenie, aby sme dostali rýchlosť. Rýchlosť ešte raz integrujeme, aby sme dostali novú pozíciu častice. Tým máme všetko vypočítané.

Vplyv externých síl môžme vypočítať v druhom alebo v treťom kroku.



Obrázek 4.2: Navrhnutý SPH model pre GPU. Kvôli dátovým závislostiam potrebujeme tri prechody nad časticami a na každý prechod jeden GPU kernel. Každý kernel má na starosti riešiť iné úlohy.

Metódu SPH, ktorá je implementovaná na grafickej karte stačí rozdeliť na tri kroky, aby sme vyriešili dátové závislosti vo výpočtoch.

4.3 Integrácia síl

Posledným krokom algoritmu SPH je integrácia zrýchlenia a rýchlosti. V simulácii tekutiny aktuálne pozície častíc sú závislé na čase. Rovnica 2.25 sa používa na vypočítanie zrýchlenia častice , rýchlosť častice dostaneme jej numerickou integráciou. Pozícia je vypočítaná integráciou rýchlosti. V tejto kapitole sú popísané metódy vhodné na numerickú integráciu, ktoré nie sú časovo náročné, ale produkujú vyhovujúce výsledky.

Implicitné a explicitné metódy

V explicitnej metóde vypočítame budúci stav systému na základe aktuálneho stavu systému v danom čase, kým v implicitnej metóde nájdeme riešenie vypočítaním rovníc, ktoré zahŕňajú aktuálny a budúci stav súčasne. Matematicky môžeme vyjadriť nasledovne. Ak aktuálny stav je $Y(t)$ a $Y(t+\Delta t)$ je budúci stav (Δt je časový krok), potom pre explicitnú metódu platí:

$$Y\left(t+\Delta t\right)=F\left(Y\left(t\right)\right) \tag{4.1}$$

Pre implicitnú metódu platí:

$$G\left(Y\left(t\right),Y\left(t+\Delta t\right)\right)=0, \tag{4.2}$$

kde hľadáme riešenie pre $Y\left(t+\Delta t\right)$.

Je jasné, že implicitné metódy potrebujú viac výpočtov a sú tažšie implementovateľné.

Implicitná Eulerova integrácia

Implicitná Eulerova schéma je skutočne semi-implicitná metóda, implicitná je len aktualizácia pozície. Semi-implicitná Eulerova metóda je založená na explicitnej Eulerovej metóde, ktorá je pravdepodobne najpoužívanejšia integračná metóda. V explicitnej Eulerovej metóde pozícia a rýchlosť sú aktualizované súčasne.

$$v_{i+1}=v_i+a_i\Delta t \tag{4.3}$$

$$r_{i+1}=r_i+v_i\Delta t \tag{4.4}$$

V semi-implicitnej Eulerovej metóde aktualizácia pozície a rýchlosti sú závislé. Aktualizácia rýchlosti je rovnaká ako v 4.3, ale na aktualizáciu pozície používame výsledok z aktualizovanej rýchlosti k predpovedaniu novej pozície.

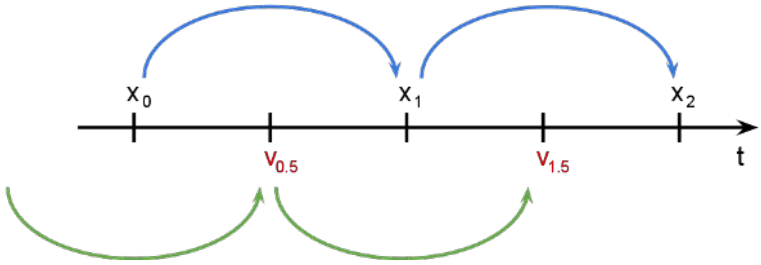
$$r_{i+1}=r_i+v_{i+1}\Delta t \tag{4.5}$$

Leap-frog integrácia

K integrovaniu času sme zvolili metódu nazývanú *leap-frog* integrátor, ktorá má presnosť druhého rádu a je veľmi stabilná:

$$x_{i+1}=x_i+v_{i+\frac{1}{2}}\Delta t, \tag{4.6}$$

$$v_{i+\frac{1}{2}}=v_{i-\frac{1}{2}}+a_i\Delta t, \tag{4.7}$$



Obrázek 4.3: Leap-frog integrátor, vodorovná čiara predstavuje čas, v je rýchlosť a x je pozícia.

kde i je časový krok.

Metóda dostala názov *Leap-frog*, lebo rýchlosť preskakuje cez pozície, a naopak. Túto schému je vidieť na obrázku 4.3 Rovnice môžeme prepísať do takého tvaru, kde sú používané len celé čísla:

$$x_{i+1} = x_i + v_i \Delta t + \frac{1}{2} a_i \Delta t \tag{4.8}$$

$$v_{i+1} = v_i + \frac{1}{2} \left(a_i + a_{i+1} \right) \Delta t \tag{4.9}$$

Leap-frog je dobrý kompromis medzi natívnou Eulerovou metódou a pokročilejšími metódami, ktoré potrebujú viac výpočtov pre každú silu. Takéto pokročilé metódy, ako napr. *Runge-kutta* tretieho rádu, boli používané v [6] pre SPH.

Kapitola 5

Realizácia

Pri realizácii časticového systému najdôležitejšie používané technológie, metódy a algoritmy boli tieto:

- platforma CUDA,
- metóda SPH,
- uniformná mriežka,
- Marching cubes,
- tessellácia,
- OpenGL a SDL.

Niektoré zo zoznamu už boli podprobnejšie rozpísané v predchádzajúcich kapitolách. V tejto kapitole sa budeme sústrediť na dôležité implementačné a realizačné časti projektu.

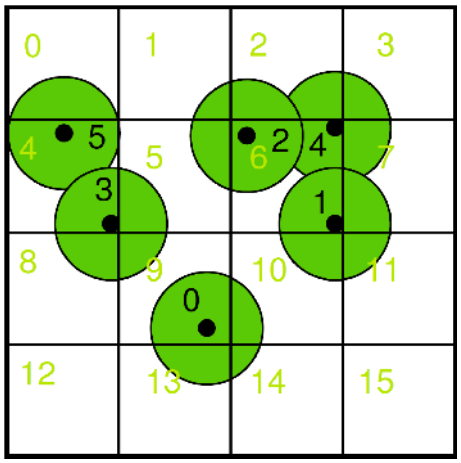
Program je implementovaný v jazyku *C++*. Dôvody pre výber jazyka *C++* boli tieto: platformová nezávislosť, podpora objektovo orientovaného programovania a mnohotvárnosti, a samozrejme rýchlosť. Veľká časť zdrojových súborov je napísaná pre grafickú kartu, ktorá počíta simuláciu, generovanie trojuholníkov a zobrazuje výsledky. Nová verzia CUDA API, presnejšie verzia 4.0, už dostala podporu pre jazyk *C++*. Vďaka tejto podpory sú zdrojové kódy čitateľnejšie a vo väčšine prípadov uľahčujú programátorskú prácu. Skoro všetky výpočty sú vykonávané na grafickej karte, od samotnej simulácie až po zobrazenie výsledkov.

Simulácia má tri pomocné kroky k urýchleniu výpočtov a prístupov k dátam, čo súvisí s uniformnou mriežkou a s tromi krokmi algoritmu SPH, viď. obrázok 5.3.

V nasledujúcich podkapitolách sa sústredíme na implementačné detaily, problémy a ich riešenie a optimalizácie.

5.1 Uniformná mriežka

Uniformná mriežka slúži na sledovanie a nájdenie susedných častíc. Táto štruktúra predstavuje najjednodušie delenie priestoru. Z pomenovania vyplýva, že mriežka je delená na rovnako veľké bunky. Do týchto buniek sú zaradené jednotlivé častice. Každá častica je priradená len do jedinej bunky. Z dôvodu, že častica môže prekrývať viac buniek, pri spracovávaní je nutné skúmať aj častice susedných buniek. Takú situáciu ukazuje obrázok 5.1.

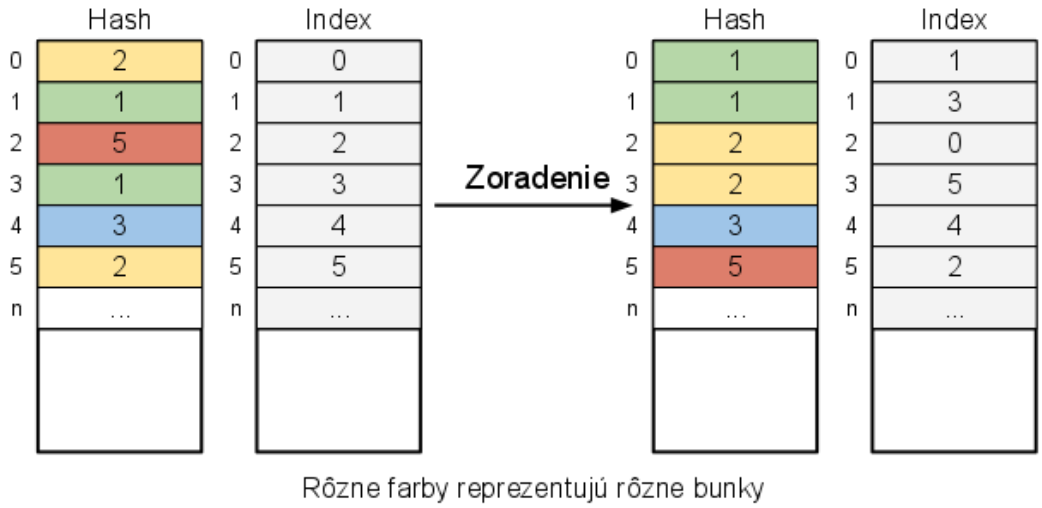


Obrázek 5.1: Prekrývanie buniek častíc v uniformnej mriežke.

Mriežka umožňuje rýchle vyhľadávanie okolitých častíc, stačí k tomu buď pozícia častice, alebo jej hash kód. Hash kód je identifikátor bunky častice. Na jednu bunku môže pripadať viac častíc, čo zamená, že budú mať rovnaké kódy. Vlastnosti častíc sú uložené do polí, takže každá častica musí mať jedinečný index.

Kroky uniformnej mriežky pri aktualizácii údajov:

- vypočítanie hash kódov častíc na základe pozícií,
- zoradenie častíc podľa buniek,
- nájdenie indexov začiatku a konca buniek.



Obrázek 5.2: Zoradenie indexov podľa hash kódov v uniformnej mriežke.

Najprv je nutné vypočítať nový hash kód každej častice na základe jej pozície. Potom nasleduje zoradenie hash kódov a indexov častíc. Na obrázku 5.2 je vysvetlený algoritmus zoradenia indexov. Ľavá strana ukazuje hash kódy a indexy častíc pred zoradením. Čísla vedľa stĺpcov reprezentujú indexy do polí. Zoradíme polia takým spôsobom, že hash kódy

budú od najmenšieho po najväčší. Keď prehodíme hash kód, synchronne prehodíme aj index častice. Nakoniec zoradené polia už nám umožňujú indentifikovať začiatok a koniec buniek. Začiatok a koniec bunky sú indexy, ktoré ukažujú odkiaľ až pokiaľ sú častice v poli, ktoré patria do rovnakej bunky. Častice, ktoré patria do rovnakej bunky budú po zoradení v poli umiestnené za sebou. Napríklad bunka 1 obsahuje dve častice s indexmi 1 a 3, teda začiatok bunky je index 0 a koniec 1.

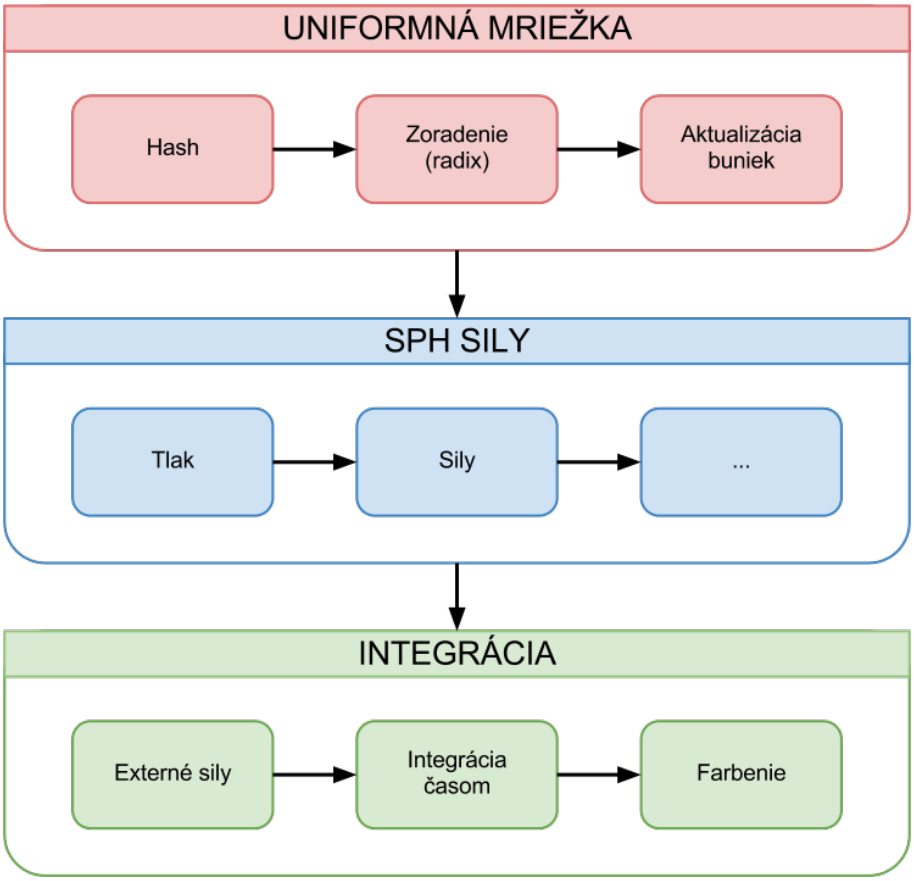
5.2 Efektívna implementácia metódy SPH

Simulačný systém simuluje kvapalinu, ktorá je založená na metóde SPH. Táto metóda bola predstavená podrobnejšie v kapitole 2.5.

Jedna iterácia algoritmu SPH je jeden časový krok, tzv. inkrementácia simulačného času. Tri hlavné kroky implementovaného algoritmu SPH sú názornené na Obr. 5.3.

Prvým krokom je aktualizácia akceleračných štruktúr (uniformnej mriežky). Celý krok je podrobnejšie popísaný v kapitole 5.1.

Druhý krok je výpočet síl v SPH. Poradie jednotlivých krokov je dané závislosťou dát medzi rôznymi výpočetmi. V SPH funkcia 2.13, ktorá slúži na výpočet síl je závislá na tlaku. To znamená, že pred výpočtom síl je nutné najprv vypočítať tlak v bode každej častice. V



Obrázek 5.3: 3 hlavné kroky simulácie: (1) Úprava mriežky (2) Výpočet síl (3) Integrácia. Každý krok obsahuje v sebe ďalšie výpočty, ktoré sú riešiteľné jediným prechodom častíc.

poslednom kroku je inkrementovaný simulačný čas a vypočítaná integrácia zrýchlenia častíc

časom. Ku pôsobiacim silám sú pripočítané externé sily, ako gravitácia, a sily z interakcie s hranicou simulačného priestoru.

Model

Model SPH je založený na modelu od Müllera [22]. Tento model používa špecializované vyhladzovacie jadrá. Má tri hlavné výpočetné kroky: výpočet hustoty, tlakovej sily, a viskóznej sily.

SPH hustota podľa rovnice 2.29:

$$\rho_i = \sum_{j \neq i} m_j W_{poly6} \tag{5.1}$$

SPH tlak podľa rovnice 2.34:

$$f_i^{pressure} = -\sum_{j \neq i} \frac{p_i + p_j}{2} \frac{m_j}{\rho_j} \nabla W_{spiky}. \tag{5.2}$$

Tlak p je možné vypočítať z hustoty pomocou tejto rovnice:

$$p = k\left(\rho - \rho_0\right). \tag{5.3}$$

SPH viskozita podľa rovnice 2.42:

$$\mathbf{f}_i^{viscosity} = \frac{\mu}{\rho_i} \sum_j \left(\mathbf{v}_j - \mathbf{v}_i\right) m_j \nabla^2 W_{viscosity}. \tag{5.4}$$

Všetky premenné a konštanty boli presne vysvetlené v kapitole 2.

Prekalkulácia

Výpočty, v ktorých sa používajú vyhladzovacie jadrá sa dajú optimalizovať pomocou prekalkulácie nemenných častí vyhladzovacích jadier. Každé jadro sa dá rozdeliť na dve časti, na konštantnú (W^{const}) bez premennej vzdialenosti $\mathbf{r}$ a na časť, ktorá je závislá na premennej $\mathbf{r}$ (W^{var}).

Vzhľadom k tomu, že v simulácii sa používa homogenná tekutina (všetky častice majú konštantnú hmotu) môžeme násobenie hmotou premiestniť pred sumu, a vyjadriť hustotu nasledovne:

$$\rho_i = m * W_{poly6}^{const} * \sum_{j \neq i} W_{poly6}^{var}, \tag{5.5}$$

$$W_{poly6}^{const} = \frac{315}{64\pi h^9}, \tag{5.6}$$

$$W_{poly6}^{var} = \left(h^2 - |\mathbf{r}|^2\right)^3. \tag{5.7}$$

Optimalizovaná verzia výpočtu tlakovej sily:

$$f_i^{pressure} = -m * \nabla W_{spiky}^{const} * \frac{1}{2} * \frac{1}{\rho_i} \sum_{j \neq i} \frac{p_i + p_j}{\rho_j} \nabla W_{spiky}^{var}, \tag{5.8}$$

$$\nabla W_{spiky}^{const} = -\frac{45}{\pi h^6}, \tag{5.9}$$

$$\nabla W_{spiky}^{var} = \frac{\mathbf{r}}{|\mathbf{r}|}\left(h - |\mathbf{r}|\right)^2. \tag{5.10}$$

Nakoniec optimalizované rovnice na výpočet viskóznej sily:

$$f_i^{viscosity} = \mu * m\frac{1}{\rho_i} * \nabla^2 W_{viscosity}^{const} * \sum_{j \neq i} \frac{\mathbf{v}_i + \mathbf{v}_j}{\rho_j} \nabla^2 W_{spiky}^{var}, \tag{5.11}$$

$$\nabla^2 W_{viscosity}^{const} = \frac{45}{\pi h^6}, \tag{5.12}$$

$$\nabla^2 W_{viscosity}^{var} = (h - |\mathbf{r}|)\,. \tag{5.13}$$

Kvôli datových závislostí pre tento model potrebujeme dva kroky iterácie nad časticami. Tlaková aj viskózna sila je závislá na hustote, preto v prvom kroku vypočítame hustotu. Po tomto kroku nasleduje výpočet tlakovej a viskóznej sily, ktoré zakombinujeme do jedného kroku.

Narvhnutá optimalizácia značne znižuje čas potrebný k výpočtom a umožňuje vyhnúť sa zbytočných prepočítaní konštantných častí rovníc.

Pseudokódy

V tejto podkapitole popíšeme pseudokódy SPH výpočtov, čo pomôže lepšie pochopiť výpočty metódy. Prvý krok metódy SPH ukazuje algoritmus 1. Je to krátky algoritmus, v cyklu akumulujeme hodnotu premennej časti vyhladzovacieho jadra a nakoniec hodnotu vynásobíme hmotou častíc a konštantnou časťou jadra.

Pretože sily tlaku a viskozity nie sú dátovo závislé, môžeme ich vypočítať v jednom iteračnom kroku, viď algoritmus 2. Tento algoritmus je už mierne komplikovanejší. Funguje podobne ako predchádzajúci, ale táto iterácia akumuluje nielen premenné časti jadier, ale aj výsledky tlakových a rýchlosťných výpočtov.

Algoritmus 1 Pseudokód pre výpočet hustôt.

```
for particles : i do
    density[i] = 0;
    for neighbourCells : cell do
        for particle in cell : j do
            r = position[i] - position[j];
            l = length(r);
            if l <= h then
                density[i] += W_poly6^var(r, h);
            end if
        end for
    end for
    density[i] *= mass * W_poly6^const;
end for
```

5.3 Zobrazenie

Užívateľ si môže vybrať z týchto typov zobrazení:

- Point Sprites (guľôčkové zobrazenie),
- Marching cubes,

Algoritmus 2 Pseudokód pre výpočet síl.

```
for particles : i do
    forces[i] = 0;
    force.pressure = 0;
    force.viscosity = 0;
    for neighbourCells : cell do
        for particle in cell : j do
            r = position[i] - position[j];
            l = length(r);
            if l <= h then
                density[i] += W^var_poly6(r, h);
                force.pressure +=
                    ((pressure[i] + pressure[j]) / density[j]) * W^var_pressure(r, h);
                force.viscosity +=
                    ((velocity[j] - velocity[i]) / density[j]) * W^var_viscosity(r, h);
            end if
        end for
    end for
    force[i] += (- mass * W^const_spiky * force.pressure) / (2 * density[i]);
    force[i] += (μ * mass * W^const_viscosity * force.viscosity) / density[i];
end for
```

- Marching cubes s tesseláciou.

Výsledky jednolivých zobrazení je možné vidieť v kapitole 6.

Point sprites

Zobrazenie metódou Point Sprites, ako už bolo popísané v podkapilole 3.1, nepotrebuje žiadne podrobnejšie vysvetlenie. Je to najjednodušia metóda, ktorápoprové sa dá implementovať na pár riadkoch.

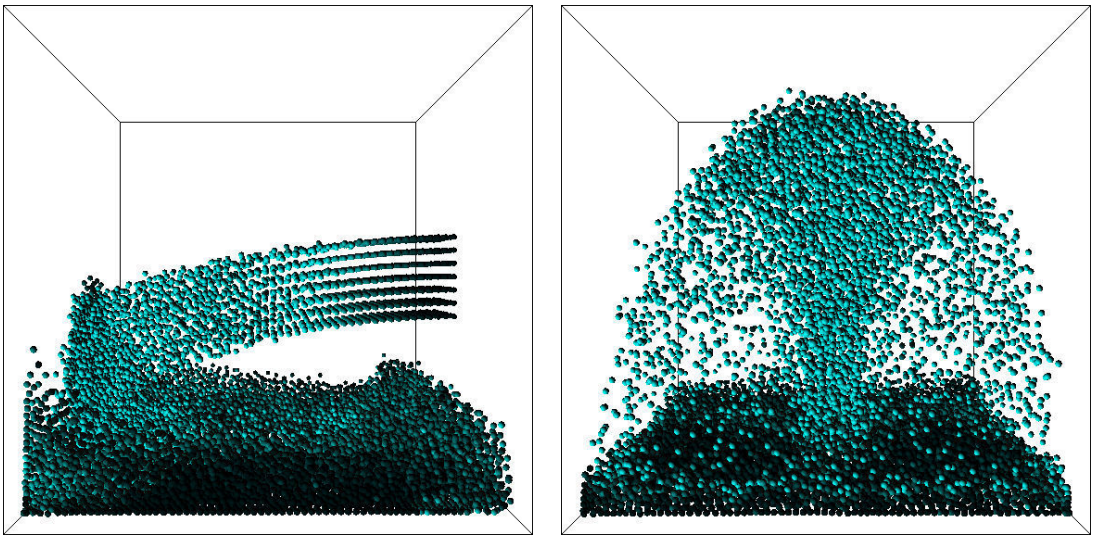
Farbenie

Po prepnutí na toto zobrazenie môžeme zapnúť dynamické farbenie častíc. Zdrojom farbenia môže byť:

- rýchlosťné pole
- silové pole (interné + externé sily)

Farby gradientov sú tieto:

- biela
- čierna → sedá
- čierna → azúrová
- modrá → biela
- HSV modrá → HSV červená



(a) Fontána púšťaná z boku.

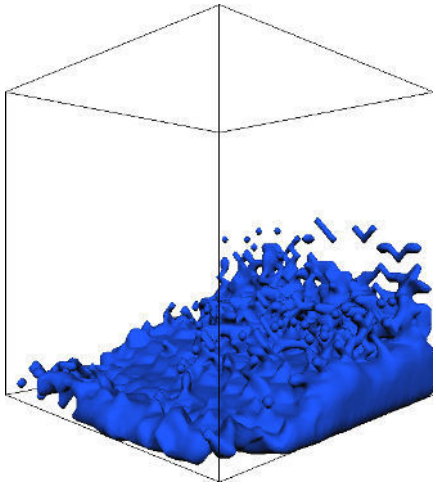
(b) Fontána púšťaná z dola.

Obrázek 5.4: Fontány zobrazené metódou Point Sprites.

U gradientov prvá farba je používaná na malé hodnoty a druhá na veľké. Napr. pri voľbe gradientu čierna → azúrová a zdroja silové pole, čierné regióny budú znamenať, že pôsobia malé sily a azúrové veľké sily.

Marching cubes

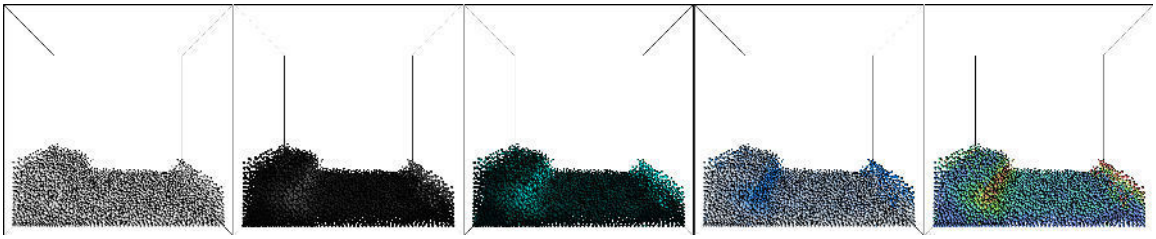
Metóda Marching cubes je ľahko paralelizovateľná, a preto bola implementovaná tak, aby algoritmus bežal s využitím GPU. Pri implementácii boli používané predpočítané look-up tabuľky, ktoré dokážu značne znížiť dobu trvania výpočtov. Používané sú tri tabuľky. Prvá tabuľka obsahuje indexy hrán, ktoré po troch tvoria trojuholníky. Tieto trojuholníky použijeme na pokrytie izoplochy. Z druhej tabuľky môžeme zistiť koľko trojuholníkov potrebujeme na pokrytie pre danú konfiguráciu. Posledná tabuľka sa používa na interpoláciu. Pomocou tejto tabuľky sme schopní zistiť, že pre určitý vrchol aké budú indexy susedných vrcholov, čo veľmi zjednodušuje prácu pri interpolovaní normálov. Pobrobnejší popis o indexácii vrcholov a indexov, a o používaní bitových vektorov obsahuje podkapitola 3.2.



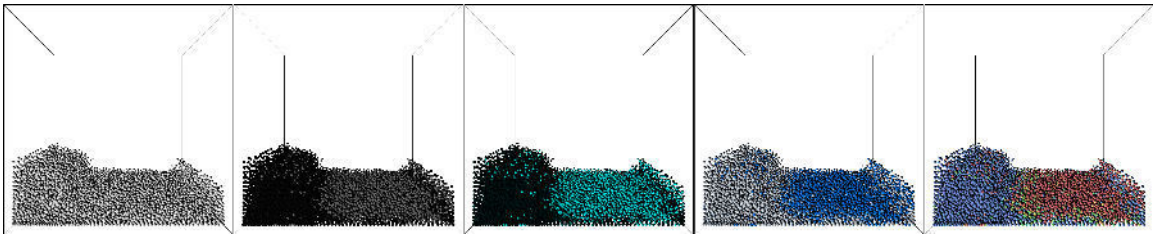
Obrázek 5.6: Tekutina generovaná metódou Marching cubes.

Kroky algoritmu:

- **Vzorkovanie.** Zistíme ktoré kocky budú tvoriť povrch izoplochy, aké sú ich konfigurácie a koľko trojuholníkov potrebujeme na pokrytie.
- **Sčítanie aktívnych kociek.** Ak ich počet je nula, netreba spusiť sa do ďalších krokov.
- **Kompresia indexov.** Indexy aktívnych kociek zapíšeme do nového pola.

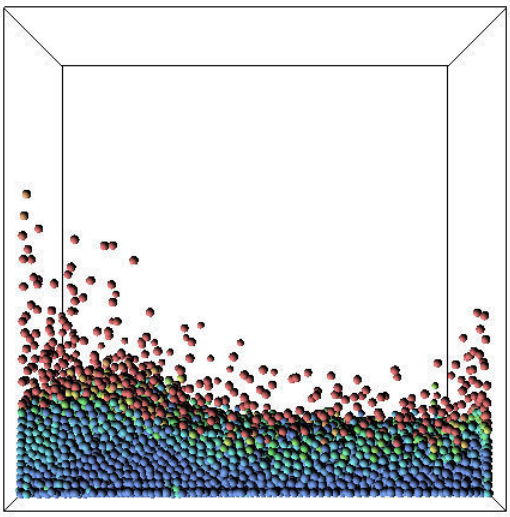


(a) Rýchlosťné pole.

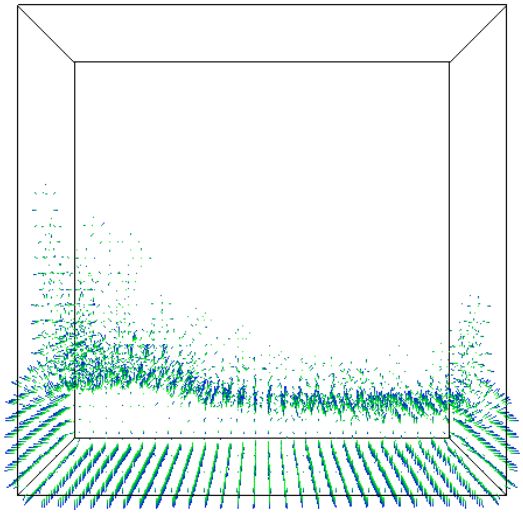


(b) Silové pole.

Obrázek 5.5: Rôzne typy gradientov: (1) biela (2) čierna → šedá (3) čierna → azúrová (4) modrá → biele (5) HSV modrá → HSV červená.



(a) Guľôčky.



(b) Normály.

Obrázek 5.7: Ukázka povrchových normálov.

- **Ščítanie trojuholníkov.**
- **Generovanie trojuholníkov.** Generované sú aj normály, ale normály vo vrcholoch každého trojuholníka pozerajú rovnakým smerom.
- **Interpolácia normálov.**

V kroku vzorkovania je používaná mriežka simulátora. Kvôli malej výkonnosti používanej grafickej karty nie je vypočítaná hustota izoplohy v každom vrcholu kocky. Namiesto toho sa používa hustota častíc v danej bunke. Samozrejme toto riešenie má aj nedostatky, ale

nie je potrebné spustiť sa do časovo náročných výpočtov. Ak máme dostupnú dostatočnú výpočetnú kapacitu na presné vyhodnotenie izoplochy častíc, môžeme hustotu vypočítať v ľubovoľnom bode:

$$g(\mathbf{x}) = \sum_i d_i f_i(|\mathbf{x} - \mathbf{r}_i|), \tag{5.14}$$

kde f_i je funkcia hustoty, d_i sú koeficienty hustoty. Pre $f_i(\mathbf{r}) = 0$ platí, ak $r > R$, kde R je efektívny polomer vplyvu hustoty častice. Funkcia $f_i(\mathbf{r})$ monotónne klesá zvýšením hodnoty $\mathbf{r}$.

Kompresiu indexov môžeme predstaviť nasledovne. Ak máme tri pole:

$$A = \{0, 1, 2, 3, 4, 5\},$$
$$F = \{1, 0, 1, 1, 0, 1\},$$
$$C = \{0, 2, 3, 5, X, X\},$$

kde pole A obsahuje pôvodné indexy, pole F príznaky obsadenosti (aktívne / pasívne) kociek a C komprimované indexy. X znamená, že tie elementy nás nezaujímajú. Pomocou polí A a F vytvoríme pole C, ktoré obsahuje indexy len aktívnych kociek. Keď v F je 0, nepridáme index z A do C.

V kroku generovania trojuholníkov vypočítame pozície na základe pretínaných hrán a interpolujeme ich podľa relatívnej hustoti vrcholov.

V poslednom kroku je nutné zpätne mapovať vrcholy susedných trojuholníkov a interpolovať ich. Bez interpolovaných normálov sa nedá spraviť hladké tienenie povrchu. Na obrázku 5.7 sú znázornené extrahované normály povrchu.

Interpolácia a tessellácia

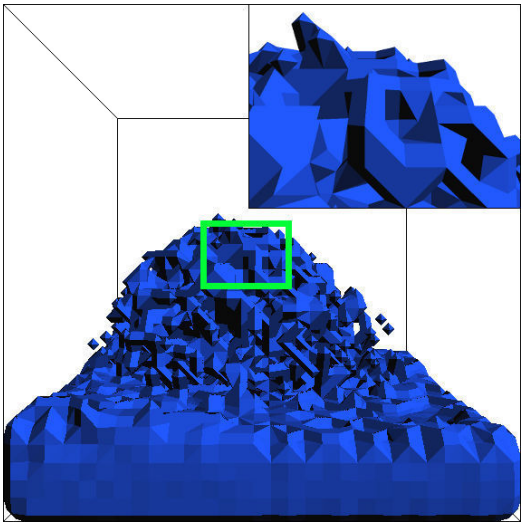
V tejto podkapilole budeme skúmať rozdiely jednolivých výsledkov, ktoré boli generované metódou Marching cubes.

Obrázok 5.8(a) ukazuje výsledky bez interpolácie normálov. Je vidieť, akým smerom sú orientované jednotlivé trojuholníky a každý je vyfarbený rovnakým odtieňom, čo veľmi zhoršuje vizuálnu kvalitu.

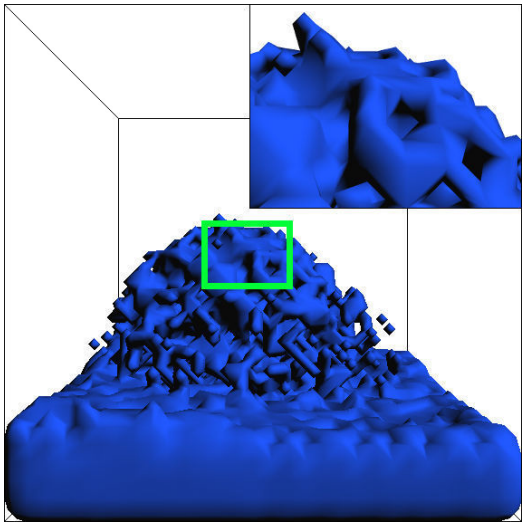
Na druhom obrázku 5.8(b) vidíme, aké sú výsledky so zapnutou interpoláciou povrchových normálov. Už nie je možné rozpoznať jednotlivé trojuholníky, sú krajšie tienené s rôznymi odtieňmi. Problém je, že ostré hrany sú stále viditlné.

Na tento problém existujú dve riešenia. Prvé je používať adaptívnu mriežku. Najprv navzorkovať objem hrubou mriežkou. Ak už poznáme aktívne kocky, tak môžeme navzorkovať objem znova s mriežkou, ktorá má už väčšie rozlíšenie. Vzvorkovať budeme len v aktívnych regiónoch, čo ušetrí veľa času. Druhá možnosť je oveľa jednoduchšia. Použijeme tesselátor grafickej karty a budeme deliť trojuholníky na menšie a nové body posunieme podľa interpolovaných normálov, aby sme dostali zaoblený porvh. Toto riešenie ukazuje obrázok 5.8(c).

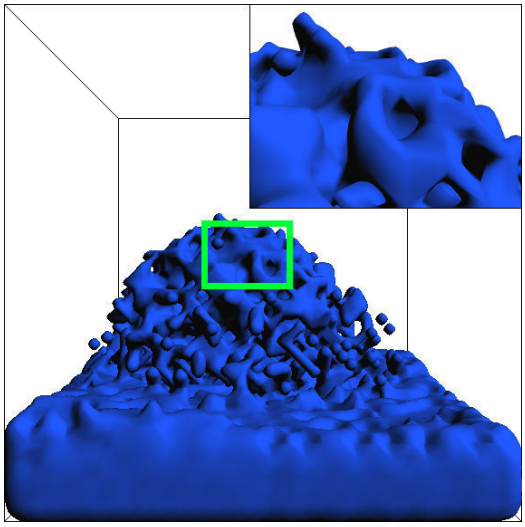
Posledný obrázok ukazuje rozdiel medzi vypnutou a zapnutou tesselláciou. Na ľavom obrázku 5.9(a) trojuholníky sú v pôvodnom stave. Na obrázku 5.9(b) sú trojuholníky delené na menšie a vytvorené nové body sú posunuté. Výpočet delenia trojuholníkov a posunutie ich vrcholov je popísané v podkapitole 3.2. Je vidieť, že na tomto obrázku povrh je už hladký.



(a) Bez interpolácie.

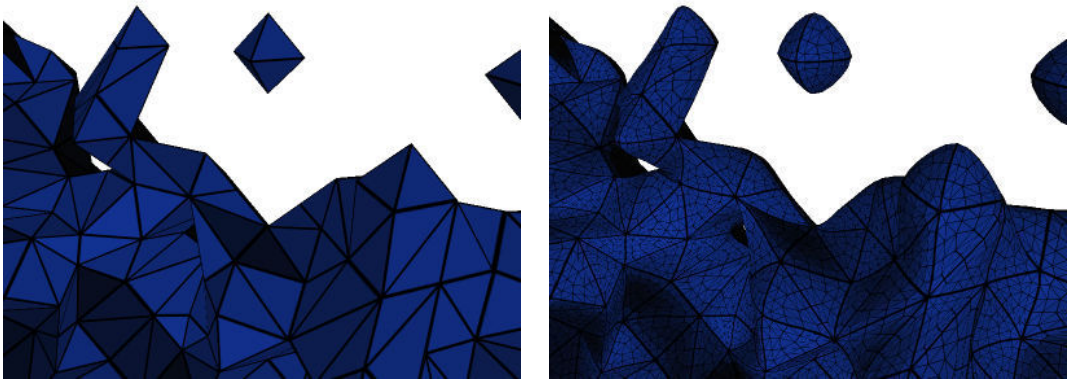


(b) Interpoláciou.



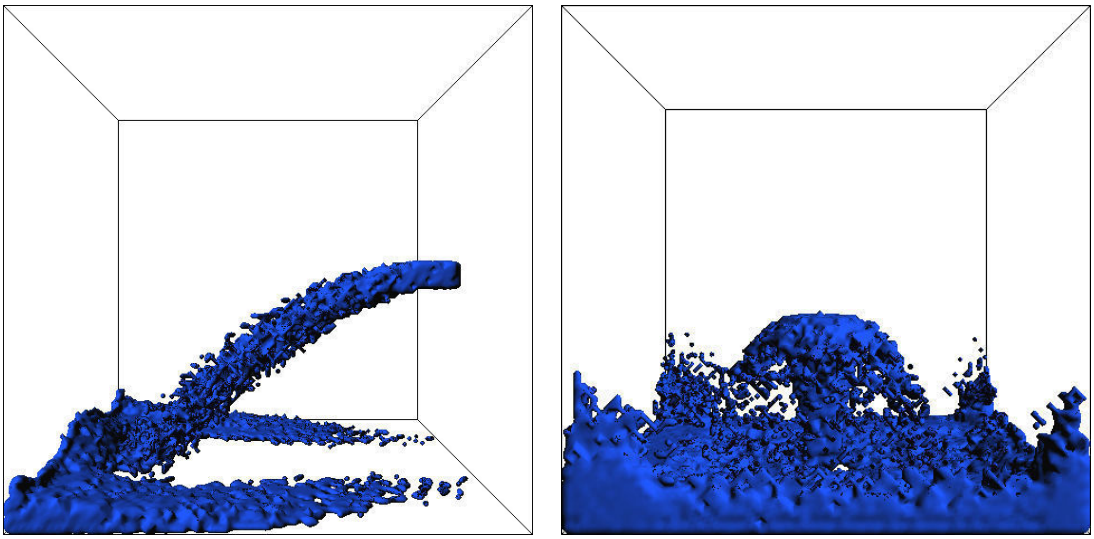
(c) Tesselláciou.

Obrázek 5.8: Ukázka interpolovaných normálov a tessellácie.



(a) Bez tessellácie. (b) Tesselláciou.

Obrázek 5.9: Ukázka trojuholníkov pri tessellácie.



(a) Fontána púšťaná z boku. (b) Fontána púšťaná z dola.

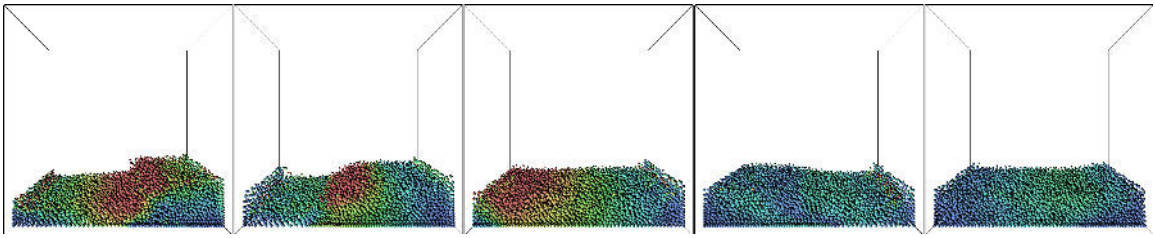
Obrázek 5.10: Fontány zobrazené metódou Marching cubes.

Kapitola 6

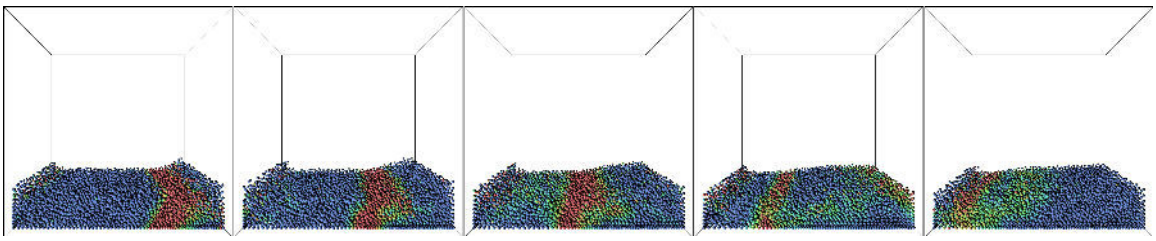
Výsledky

Táto kapitola obsahuje zhrnutie dosiahnutých výsledkov tejto práce a práce do budúcna.

Program je implementovaný v programovacom jazyku C++. Komponenty systému boli navrhnuté vysokou mierou znovupoužiteľnosti. Pri implementácii boli používané rôzne typy bufferov (na skalárne hodnoty, vektory, adť.) pre rôzne pamäťové miesta (pamäť CPU a GPU). Aby nebolo nutné pre všetky typy implementovať špeciálne buffery a managery bufferov, boli tieto komponenty implementované vysokou abstrakciou, čo zjednodušuje prácu programátora i pochopenie samotného programu. Táto koncepcia platí aj pre ostatné časti systému.



(a) Rýchlosťné pole.



(b) Silové pole.

Obrázek 6.1: Prietoky v rýchlostnom a v silovom poli použitím metódy Point Sprites.

Obrázky generované programom sa nachádzajú v kapitole 5. V programe je možné zapnúť fontánu, vid obrázok 5.10. Fontána tečie buď z boku, alebo zo spodu. Pomocou otáčania kocky môžeme spraviť, aby tekunina tiekla napr. z hora. Fontána má dve nastaviteľné parametre: počet a rýchlosť častíc, ktoré vystupujú z nej za jednu časovú jednotku (časový krok).

Smer gravitácie sa dá meniť tiež otáčaním kocky, takým spôsobom sme schopní napr. rozhojdať tekutinu.

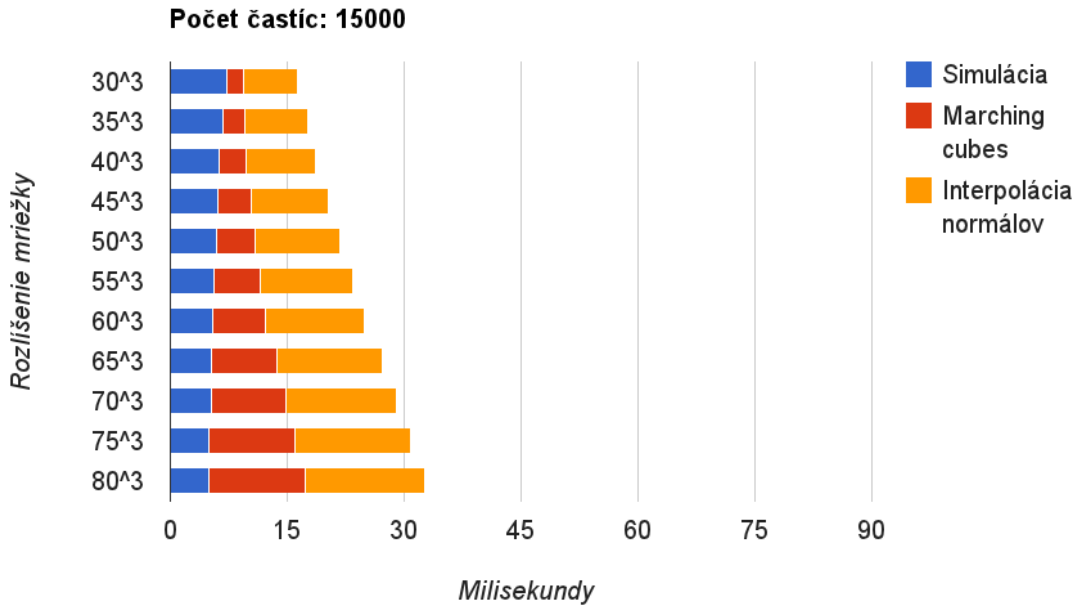
Pri voľbe zobrazenia Point Sprites je možné sledovať prietok rýchlostí a síl a ich postupné vyhladenie na začiatku tečúcej tekuniny. Tieto javy ukazuje obrázok 6.1.

6.1 Testovanie

Algoritmy implementované pre beh v GPU boli tiež napísané v C++ a použitím platformy CUDA, s CUDA API verziou 4.0. Program bol testovaný na grafickej karte NVidia GeForce 540M, ktorá má 96 jadier. Používaným operačným systémom bol Ubuntu 11.04 (64bit) s linuxovým jadrom 2.6.38-13-generic.

Výsledky testov sú zobrazené tabuľkovou formou (tabuľky 6.1, 6.2 6.3, 6.4) aj graficky (obrázky 6.2, 6.3, 6.4 a 6.5), kvôli lepšiemu pochopeniu.

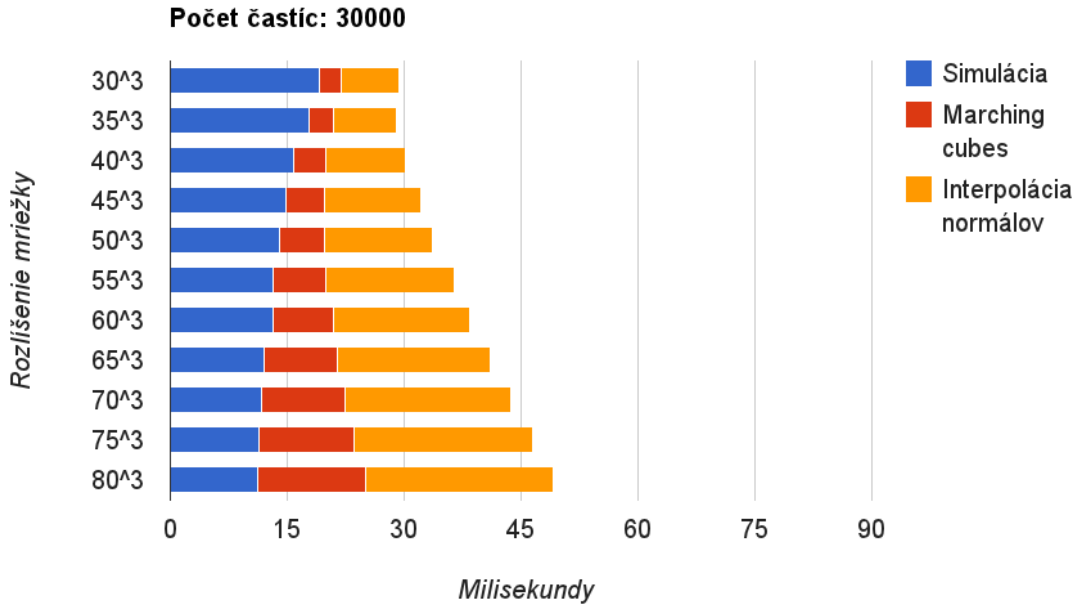
Maximálna hranica počtu častíc, pri ktorom simulácia pobeží v reálnom čase je okolo 60000 pri rozlíšení mriežky 80×80×80. V prípade, že zapneme zobrazovaciu metódu Marching cubes s interpoláciou, výkon pri daných parametroch extrémne poklesne, viď obrázok 6.5.



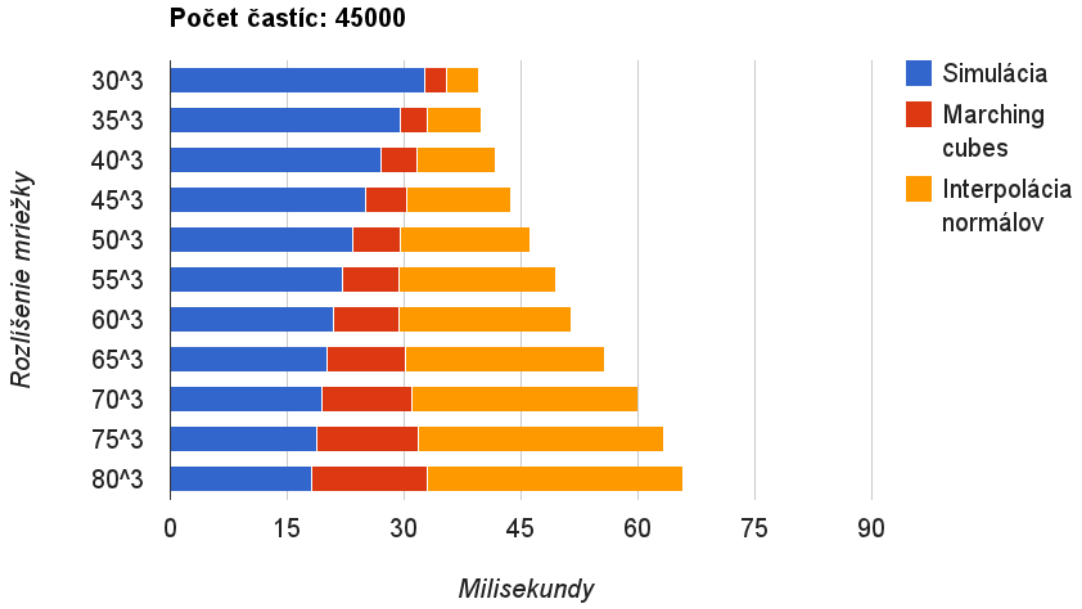
Obrázek 6.2: Výsledky testovania s použitím 15000 častíc.

Z testov je vidieť, že postupným zvýšením rozlíšenia simulačnej mriežky sa zníži doba výpočtu simulácie. To je pôsobené tým, že v priemere na jednu bunku padne menej častíc pri menšom rozlíšení. Tým, že zvýšime rozlíšenie, zvýšime aj výpočetné nároky na údržbu mriežky (hešovanie a zoradenie častíc, aktualizácie buniek). Kvôli tomu, že metóda Marching cubes používa mriežku simulácie, zvýšime čas potrebný k výpočtom, ale tým pádom zvýšime aj kvalitu výsledkov, teda hladkosť povrchu.

Samotná metóda Marching cubes nie je citlivá na počet používaných častíc. To je z dôvodu implementácie, ako už bolo popísané v podkapilole 5.3. Kvôli malej výkonnosti používanej grafickej karty nebolo možné v každom bode mriežky počas vzorkovania vypočítať hustotu izoplochy, preto boli používané hodnoty hustôt buniek. Toto riešenie funguje

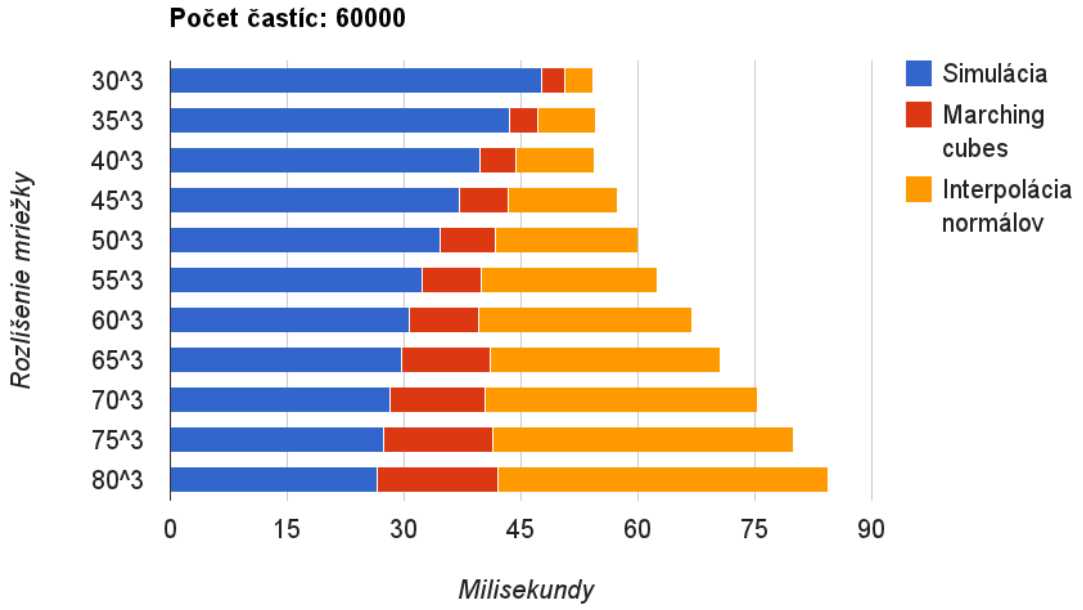


Obrázek 6.3: Výsledky testovania s použitím 30000 častíc.



Obrázek 6.4: Výsledky testovania s použitím 45000 častíc.

veľmi rýchle a pomocou relatívnej hustoty sa dajú interpolovať vrcholy na hranách kociek. Negatívna strana toho riešenia je menej presný vyextrahovaný povrch. Veľká výhoda pri



Obrázek 6.5: Výsledky testovania s použitím 60000 častíc.

generovaní výsledkov je, že táto metóda nie je závislá na finálnom rozlíšení zobrazenia, ako napr. metóda Ray casting.

6.2 Práce do budúcna

V tejto kapitole krátko popíšeme možné vylepšenia a optimalizácie narvhnutého a implementovaného systému.

Čo sa týka simulácie z fyzikálnej strany, chýba implementácia porvchového napätia. Táto sila má na tekutinu len malý vplyv, ale ak chceme dosiahnuť vyššiu kvalitu výsledkov simulácie, tak výpočty fyzikálneho modelu by mali vypočítať pôsobenie tejto sily tiež. Zo strany optimalizácie treba nájsť a predpočítať všetky časti algoritmu, ktoré sú vypočítané zbytočne viackrát. Ďalej by sa dalo optimalizovať siahanie do globálnej pamäte, napr. používaním textúr, ktoré sú kešované.

Zobrazenie metódou Marching cubes potrebuje značné vylepšenie, čo na používanej grafickej karte nebolo realizovateľné. Sem patrí extrahovanie povrchu tekutiny, ako už bolo popísané v kapitole 5.3; v bodoch mriežky vypočítať hustotu izoplochy. Ďalšia práca je implementovanie adaptívnej mriežky, čo bude generovať hladký povrch a zachytí aj malé kvapky tekutiny, ktoré pri malom rozlíšení mriežky nie sú detekované.

Testovanie programu na výkonnej grafickej karte by bolo veľmi zaujímavé, simulovať tekutinu, ktorá sa skladá z 100 alebo 500 tisíc častíc. Nevýhoda metódy SPH je, že potrebuje veľké množstvo častíc, aby dosiahla hodnoverné výsledky.

Ako je vidieť z výsledkov testovania interpolácia normálov potrebuje najviac času. To je z dôvodu spätného mapovania normálov a z redundacie výpočtov. Implementovaný algoritmus je veľmi jednoduchý. Problém je, že optimalizovať už nie je tak jednoduché.

	Počet častíc: 15000					
RM	30^3	35^3	40^3	45^3	50^3	55^3
SIM	7.273	6.801	6.318	6.175	5.86	5.665
MC	2.147	2.783	3.43	4.184	4.971	5.919
NI	6.979	8.116	8.84	9.936	10.889	11.801
Spolu	16.399	17.7	18.588	20.295	21.72	23.385
RM	60^3	65^3	70^3	75^3	80^3	
SIM	5.401	5.35	5.238	5.027	4.95	
MC	6.88	8.332	9.562	10.95	12.39	
NI	12.61	13.518	14.188	14.89	15.32	
Spolu	24.891	27.2	28.988	30.867	32.66	

Tabulka 6.1: Výsledky testovania boli merané v milisekundách pri rôznych rozlíšeniach mriežky. SIM znamená čas potrebný k simulácii, MC je Marching cubes, NI je interpoláca normálov. Tabuľka obsahuje dáta testovania s použitím 15000 častíc.

	Počet častíc: 30000					
RM	30^3	35^3	40^3	45^3	50^3	55^3
SIM	19.16	17.75	15.75	14.84	13.97	13.25
MC	2.738	3.152	4.178	5.005	5.733	6.723
NI	7.471	8.158	10.222	12.305	13.937	16.527
Spolu	29.369	29.06	30.15	32.15	33.64	36.5
RM	60^3	65^3	70^3	75^3	80^3	
SIM	13.11	12.1	11.76	11.45	11.24	
MC	7.778	9.319	10.7	12.1	13.76	
NI	17.562	19.691	21.21	22.97	24.13	
Spolu	38.45	41.11	43.67	46.52	49.13	

Tabulka 6.2: Výsledky testovania boli merané v milisekundách pri rôznych rozlíšeniach mriežky. SIM znamená čas potrebný k simulácii, MC je Marching cubes, NI je interpoláca normálov. Tabuľka obsahuje dáta testovania s použitím 30000 častíc.

	Počet častíc: 45000					
RM	30^3	35^3	40^3	45^3	50^3	55^3
SIM	32.68	29.57	26.99	25.01	23.34	22.09
MC	2.708	3.466	4.606	5.25	6.136	7.25
NI	4.216	6.854	10.174	13.419	16.674	20.05
Spolu	39.604	39.89	41.77	43.68	46.15	49.39
RM	60^3	65^3	70^3	75^3	80^3	
SIM	20.96	20.13	19.4	18.81	18.2	
MC	8.462	10.04	11.54	13.04	14.74	
NI	21.958	25.5	29.03	31.41	32.83	
Spolu	51.38	55.67	59.97	63.26	65.77	

Tabulka 6.3: Výsledky testovania boli merané v milisekundách pri rôznych rozlíšeniach mriežky. SIM znamená čas potrebný k simulácii, MC je Marching cubes, NI je interpoláca normálov. Tabuľka obsahuje dáta testovania s použitím 45000 častíc.

	Počet častíc: 60000					
RM	30^3	35^3	40^3	45^3	50^3	55^3
SIM	47.67	43.53	39.75	37.08	34.61	32.37
MC	2.911	3.644	4.568	6.239	7.018	7.553
NI	3.642	7.336	10.132	14.091	18.352	22.567
Spolu	54.223	54.51	54.45	57.41	59.98	62.49
RM	60^3	65^3	70^3	75^3	80^3	
SIM	30.72	29.75	28.16	27.31	26.57	
MC	8.883	11.33	12.22	14.01	15.54	
NI	27.307	29.44	34.99	38.64	42.37	
Spolu	66.91	70.52	75.37	79.96	84.48	

Tabulka 6.4: Výsledky testovania boli merané v milisekundách pri rôznych rozlíšeniach mriežky. SIM znamená čas potrebný k simulácii, MC je Marching cubes, NI je interpoláca normálov. Tabuľka obsahuje dáta testovania s použitím 60000 častíc.

Posledná vec je vytvorenie kvalitnejšieho zobrazenia, ktorá podporuje aj priesvitnosť, refrakciu a reflexiu. Tieto vlastnosti sú typické pre tekutiny, a nutné k tomu, aby sa javili čo najrealistickejšie. Metóda Ray casting podporuje všetky tieto vlastnosti. Nedostatok metódy je, že v porovnaní s Marching cubes je oveľa pomalejšia. Zrýchliť sa dá napr. s Depth peelingom, čo bolo popísané v podkapitole 3.3.

Kapitola 7

Záver

Výsledkom tejto práce je funkčný simulátor kvapalín, ktorý je založený na fyzickom modeli SPH. Kvapaliny tvoria dôležitú časť nášho prostredia v ktorom žijeme, preto je ich realistická simulácia a zobrazenie veľmi dôležité, a to nie len na používanie v počítačových hrách. Používajú sa aj pri biomedikálnych simuláciach, napr. na simulovanie krvného prietoku v žilách.

Program umožňuje užívateľovi vybrať z rôznych typov zobrazení. Tieto zobrazenia sú metóda Point Sprites a metóda Marching cubes. Čo je veľmi dôležité, užívateľ môže so interagovať systémom, zapnúť fontánu, meniť jej parametre, alebo pomiešat kvapalinu otáčaním simulačnej kocky.

Experimenty zahŕňali okrem simulácie a implementácie zobrazení aj ich urýchlenie pomocou grafickej karty, ktorú je možné programovať na riešenie všeobecných problémov.

V predchádzajúcich kapitolách boli diskutované nedostatky implementovaných metód a algoritmov, a ich teoretické riešenie a pokračovanie.

Aké pozitíva mi dal tento projekt? Najdôležitejšie je, že som rozšíril moje znalosti, skúsenosti a vyskúšal veci, ktoré ma už dávnejšie zaujímali. Prečítal som veľa materiálov, ktoré obsahovali aj iné problémy a ich riešenia, kvôli čomu som mohol spoznať veľa zaujímavých veci z oblasti počítačovej grafiky. Bolo zaujímavé vyhladať z materiálov relevantné informácie, pochopiť ich a pokúsiť sa implementovať ich. Ďalej som mal možnosť vyskúšať platformu CUDA, čo ja považujem v dnesnej dobe za dobrú skúsenosť. Okrem toho som mal možnosť sa zdokonaliť v jazyku C++ a naprogramovať *template* triedy. Hoci *STL* jazyka C++ už používam dávnejšie, vlastné triedy som predtým nenapísal.

Práca bola zábavná a zaujímavá. Plány na pokračovanie už mám a budem rád, keď sa môžem znova pustiť riešenia problémov.

Literatura

[1] A Quick Introduction to Subdivision Surfaces, [online].
`http://www.holmes3d.net/graphics/subdivision/`.

[2] Quad Tessellation with OpenGL 4.0, [online]. `http://prideout.net/blog/?p=49`.

[3] CASTEP Visualisation using Jmol, [online].
`http://www.tcm.phy.cam.ac.uk/~mjr/vis/vis_jmol.html`, 2007.

[4] OpenGL 4 Tessellation, [online].
`http://codeflow.org/entries/2010/nov/07/opengl-4-tessellation`, 2010.

[5] CUDA C Programming Guide, [online].
`http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf`, 2011.

[6] A. Ferrari, M. Dumbser, E. F. Toro, A. Armanini: A new 3D parallel SPH scheme for free surface flows. In *Computers and Fluids*, ročník 38, 2009, s. 1203–1217.

[7] A. Vlachos, J. Peters, C. Boyd, J. L. Mitchell: Curved PN triangles. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, 2001, iSBN 1-58113-292-1.

[8] Boubekeur, T.: *GPU Pro: Advanced Rendering Techniques*. A K Peters, Ltd., 2010, iSBN 978-1568814728.

[9] Bunnell, M.: Adaptive Tessellation of Subdivision Surfaces with Displacement Mapping, [online].
`http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter07.html`, 2009.

[10] F. Colin, R. Egli, F.Y. Lin: Computing a null divergence velocity field using smoothed particle hydrodynamics. 2005.

[11] G. R. Liu, M. B. Liu: *Smoothed particle hydrodynamics: a meshfree particle method*. World Scientific, 2003, iSBN 9-812384-56-1.

[12] G. R. Liu, M. B. Liu: Smoothed Particle Hydrodynamics (SPH): an Overview and Recent Developments. 2010.

[13] Hassani, S.: *Mathematical Methods*. Springer; 2nd ed. edition, 2008, iSBN 03-87095-03-9.

[14] I. Cantlay: *High-Speed, Off-Screen Particles, GPU Gems 3, Chapter 23.* Addison-Wesley, 2007, iSBN 321515269.

[15] J. Hongbin, D. Xin: On criterions for smoothed particle hydrodynamics kernels in stable field. In *Journal of Computational Physics*, ročník 202, 2005, s. 699–709.

[16] J. J. Monaghan: Smoothed Particle Hydrodynamics. In *Annual review of astronomy and astrophysics*, ročník 30, 1992, s. 543–574.

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

[18] K. Crane, I. Llamas, S. Tariq: *Real-Time Simulation and Rendering of 3D Fluids, GPU Gems 3, Chapter 30.* Addison-Wesley, 2007, iSBN 321515269.

[19] L. Bavoil, K. Myers: *Order independent transparency with dual depth peeling.* NVIDIA OpenGL SDK, 2008.

[20] M. Desbrun, M. Gascuel: Smoothed particles: A new paradigm for animating highly deformable bodies. *Eurographics Workshop on Computer Animation and Simulation (EGCAS)*, 1996: s. 61–76.

[21] M. Kelager: *Lagrangian Fluid Dynamics Using Smoothed Particle Hydrodynamics.* Dizertační práce, 2006.

[22] M. Müller, D. Charypar, M. H. Gross: *Particle-Based Fluid Simulation for Interactive Applications.* SCA '03 Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, 2003, iSBN 1-58113-659-5.

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

[24] P. Bourke: Polygonising a scalar field, [online]. `http://paulbourke.net/geometry/polygonise/`, 1994.

[25] P. Kipfer, M. Segal, R. Westermann: Uber-Flow: A GPU-based particle engine. *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, 2004: s. 115–122.

[26] P. W. Cleary, M. Prakash: Smooth particle hydrodynamics: status and future potential. In *Progress in Computational Fluid Dynamics, an International Journal*, ročník 7, 2004, s. 70–90.

[27] R. Geiss : *Generating Complex Procedural Terrains Using the GPU, GPU Gems 3, Chapter 1.* Addison-Wesley, 2007, iSBN 321515269.

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

[29] T. Boubekeur, M. Alexa: Phong Tessellation. In *Proceedings of ACM SIGGRAPH Asia 2008*, ročník 27, 2008, s. 1–5.

[30] W. T. Reeves: Particle Systems-a Technique for Modeling a Class of Fuzzy Objects. In *ACM Transactions on Graphics (TOG)*, ročník 2, 1983, s. 91–108.

[31] Wikipedia: Newton's laws of motion, [online].
http://en.wikipedia.org/wiki/Newton's_laws_of_motion, 2011.

[32] Wikipedia: Volume ray casting, [online].
http://en.wikipedia.org/wiki/Volume_ray_casting, 2012.

Příloha A

Obsah DVD

- *program* – Súbory k programu
  - *bin* – spustiteľný program (Linux x64)
  - *shader* – zdrojové súbory shaderov
  - *src* – zdrojové súbory
- *doc* – Dokumentácia
  - *tex* – TEX súbory
  - `technicka_zprava.pdf` – Technická zpráva
- *lib* – Používané knižnice
  - *cuda* – NVidia CUDA Toolkit 4.2.9
  - *sdl* – Simple Directmedia Layer 1.2.15
- *video* – Demonstračné videá
- `README`

Příloha B

Manual

Program je protrebné preložit pred spustením, príkaz `make` nám vyrobí spustiteľný súbor. Je nutné mať nakonfigurované CUDA Toolkit a správne nastavené cesty. Make vytvorí spustiteľný súbor, čo môžeme spustiť s príkazom `make run`.

Na ovládanie programu je možné používať myš a klávesové skratky.

Klávesové skratky:

Esc Vypnutie programu

F1 Zobrazenie Point Sprites

F2 Zobrazenie Marching cubes (základný)

F2 Zobrazenie Marching cubes (povrchové normály)

F4 Zobrazenie Marching cubes (tesselovaný)

F5 Zobrazenie Marching cubes (tesselované trojuholníky)

F6 Zobrazenie Ray casting

R Kompletné zastavenie/spustenie simulácie a generovania zobrazení

E Spustenie a zastavenie integrovania v simulácii

A Spustenie a vypnutie fontány

X Prepnutie pozície fontány

C Inicializácia pozícií častíc

I Zapnutie/vypnutie interpolovania normálov (funguje s Marching cubes)

N Zapnutie/vypnutie vykreslenia normálov (funguje s Marching cubes)

D Zapnutie/vypnutie dynamického prefarbenia (funguje s Point Sprites)

G Zmena gradientu farbenia (funguje s Point Sprites)

S Prepnutie zobrazenia medzi silovými a tlakovými poľami (funguje s Point Sprites)

W Inkrementácia faktoru premiestnenia trojuholníkov pri tessellácii

Q Dekrementácia faktoru premiestnenia trojuholníkov pri tessellácii

Ctrl + **W** Inkrementácia tessellačných úrovní

Ctrl + **Q** Dekrementácia tessellačných úrovní

F Inkrementácia rýchlosti častíc fontány

Ctrl + **F** Dekrementácia rýchlosti častíc fontány

P Inkrementácia počtu častíc fontány

Ctrl + **P** Dekrementácia počtu častíc fontány

Myš:
Ľavé tlačítko slúži na otáčanie, pravé na priblíženie.