UNIVERZA V MARIBORU

FAKULTETA ZA ELEKTROTEHNIKO,

RAČUNALNIŠTVO IN INFORMATIKO

Jernej Haložan

ALGORITEM ZA HIDRODINAMIKO ZGLAJENIH DELCEV V 2D PROSTORU

Magistrsko delo

Maribor, december 2018

ALGORITEM ZA HIDRODINAMIKO ZGLAJENIH DELCEV V 2D PROSTORU

Magistrsko delo

Študent:	Jernej Haložan
Študijski program:	Študijski program 2. stopnje
	Računalništvo in informacijske tehnologije (MAG)
Mentor:	doc. dr. Niko Lukač, mag. inž. rač. in inf. tehnol.

ZAHVALA

Zahvaljujem se mentorju doc. dr. Niko Lukaču za pomoč in vodenje pri magistrski nalogi.
Posebna zahvala tudi staršem, ki so mi omočili študij.

Algoritem za hidrodinamiko zglajenih delcev v 2D prostoru

Ključne besede: algoritem, hidrodinamika zglajenih delcev, simulacija tekočin, računalniška simulacija, interpolacija

UDK: 004.94.021(043.2)

Povzetek

V tem magistrskem delu obravnavamo algoritem za hidromehaniko zglajenih delcev v 2D prostoru. Algoritem je bil sprva razvit za probleme na področju astrofizike, kasneje pa se je uveljavil kot algoritem za simulacije tekočin. Glavna lastnost algoritma je interpolacija veličin simuliranega fizikalnega pojava s sistemi delcev. S tem se razlikuje od algoritmov, ki simulirajo fizikalne pojave s topološkimi mrežami. Predstavimo teoretično podlago algoritma v 2D prostoru in izvedbo na praktičnem primeru simulacije tekočine v 2D prostoru.

Smoothed particle hydrodynamics algorithm in 2D space

Key words: algorithm, smoothed particle hydrodynamics, fluid simulation, compter simulation, interpolation

UDK: 004.94.021(043.2)

Abstract

In this master's thesis we present smoothed particle hydrodnymics (SPH) algorithm in 2D space. SPH was originaly developed for problems in astrophysics and later gained traction in fluid simulation comunity. The main property of the algorithm is the smoothing of physical quantities of the underlying simulated problem with system of particles. This is also the main difference between SPH and other set of algorithms which solve simulation problem with mesh topology. In this thesis we will provide theoretical description of the algorithm in 2D space and practiclal demostration of 2D fluid simulation.

KAZALO

1 UVOD ... 1

2 SORODNA DELA ... 4

3 METODA SPH ... 9

3.1 Teoretična podlaga ... **12**

3.2 Definiranje delcev in gladitvenega jedra okolice ... **14**

3.3 Gostota, viskoznost in pritisk ... **20**

3.4 Časovna integracija ... **23**

3.4.1 Delno-implicitna Eulerjeva metoda ... 23

3.4.2 Verletova integracija s hitrostjo ... 24

3.4.3 Metoda žabjega koraka ... 24

4 IMPLEMENTACIJA ... 26

4.1 Iskanje sosedov ... **27**

4.2 Zaznavanje trkov in obravnavanje mej ... **28**

4.3 Parametri simulacije ... **30**

4.3.1 Parametri tekočin ... 30

4.3.2 Velikost gladitvenega jedra ... 31

4.3.3 Določitev časovni koraka ... 33

4.3.4 Implementacija algoritma ... 33

5 REZULTATI ... 39

6 SKLEP ... 42

LITERATURA ... 43

KAZALO SLIK

Slika 1: Ilustracija Langrangeove opisa toka z delci. 7
Slika 2: Ilustracija Eulerjevega opisa toka z mrežo. 7
Slika 3: Gladitveno jedro metode SPH. 11
Slika 4: Grafi funkcije, gradienta in Laplaceovega operatorja gladitvenega jedra poly6. 15
Slika 5: Grafi funkcije, gradienta in Laplaceovega operatorja gladitvenega jedra spiky. 17
Slika 6: Grafi funkcije, gradienta in Laplaceovega operatorja gladitvenega jedro za viskoznost 20
Slika 7: Ilustracija metode žabjega preskoka 25
Slika 8: Potek aglortma za simulacijo po metodi SPH. 26
Slika 9: Primer prevelikega gladitvenega jedra. 32
Slika 10: Primer premajhnega gladitvenega jedra. 32
Slika 11: Primer ustreznega gladitvenega jedra. 32
Slika 12: Diagram UML izvedenega programa 34
Slika 13: Zasnova grafičnega vmensika z izvedeno metodo SPH v začetnem stanju 35
Slika 14: Psevdokod zasnovanega algoritma za simulacijo vode z metodo SPH. 36
Slika 15: Implementacija računanja gostote 37
Slika 16: Implementacija računanja sil 38
Slika 17: Implementacija integracije in obravnavanja mej 38
Slika 18: Čas računanja gostote v izvedbi metode SPH 40
Slika 19: Čas računanja sil v izvedbi metode SPH 40
Slika 20: Čas integracije in obravnavanja mej v izvedbi metode SPH 41

KAZALO TABEL

Tabela 1: Viskoznosti in gostote nekaterih snovi. 18
Tabela 2: Lastnosti skupne vsem tekočinam. 31
Tabela 3: Parametri za simulacijo vode. 31
Tabela 4: Izmerjeni časi računanja posameznih delov izvedbe algoritma 39

1 UVOD

Tekočine igrajo pomembno vlogo v vsakdanjem življenju. Od zraka, ki ga dihamo, do oceanov, ki zavzemajo tri četrtine zemeljskega površja. Tekočine nimajo svoje oblike, saj zavzemajo obliko površja nad katerim se nahajajo. Lastnosti tekočin lahko pripišemo kapljevinam, plinom, plazmi in nekaterim trdninam (npr. plastika). Med pojave z lastnostjo tekočin uvrščamo vremenske tokove, oceanske tokove ali preprosto natakanje vodo v kozarec.

Na prvi pogled so videti našteti pojavi enostavni, vendar ostaja računalniška simulacija tekočin kompleksen problem. Teoretični opis dinamike tekočin ima dolgo zgodovino, vendar še zmeraj ostajajo odprti zelo kompleksni problemi na tem področju. Kljub temu so bile postopoma razvite različne metode, ki omogočajo realistične simulacije in animacije tekočin. S tem izboljšajo avtentičnost raznih interaktivnih aplikacij, kot npr.: medicinski simulatorji (npr. simulacija toka tekočin v človeškem telesu), simulatorji fizikalnih pojavov (npr. simulacija toka reke), računalniške igre (npr. realno-časovna vizualizacija vode), vizualni efekti v Hollywoodskih filmih [1] in v novejših aplikacijah navidezne resničnosti.

Mehanika tekočin je veda fizike, ki se ukvarja s preučevanjem gibanja in deformacije tekočin. Deli se na hidrostatiko (angl. fluid statics) in hidrodinamiko (angl. fluid hydrodynamics). V prvo skupino sodijo mirujoče tekočine, v slednjo skupino pa gibljive tekočine, ki se s časom spreminjajo. Mehanika tekočin spada v širše področje mehanike kontinuumov, ki obravnava gibanje in deformacije materialov kot kontinuumov, ne da bi se osredotočala na njihovo notranjo zgradbo.

V okviru magistrske naloge bomo obravnavali simulacije tekočin z algoritmom hidrodinamike zglajenih delcev (angl. smoothed particle hydrodynamics - SPH) oz. krajše z metodo SPH. Metoda SPH numerično aproksimira enačbe dinamike tekočin z množico delcev oz. točk. Temelji na Lagrangeovem opisu tokovnega polja, kar pomeni da ni treba v naprej definirati topološke mreže simuliranega pojava. Zato jo tudi uvrščamo med ti. brezmrežne metode (angl. meshfree methods). Bistvo Lagrangeovega opisa tokovnega polja in metode SPH so neodvisni delci, katerim pripišemo lastnosti iz katerih

aproksimiramo veličine za fizikalne enačbe dinamike tekočin.Metoda SPH se uvršča na področje računalniške dinamike tekočin (angl. computational fluid dynamics - CFD). CFD je veda, ki z numeričnimi metodami in podatkovnimi strukturami analizira in rešuje probleme toka tekočin. CFD je dobro raziskano področje z dolgo zgodovino. Začetek področja CFD lahko štejemo iznajdbo enačb Navier-Stokes v 19. stoletju. Omenjene enačbe predstavljajo osnovo za formulacijo skoraj vseh problemov na področju CFD. Skupino tekočin v grobem razdelimo v dve manjši skupini – na kapljevine in pline. V fiziki poznamo enačbe Navier-Stokes, ki opisujejo gibanje viskoznih tekočin in plinov. Gre za nelinearne parcialne diferencialne enačbe, ki opisujejo interakcijo pritiska, temperature, gostote in premikajoče se tekočine. Enačbe sta neodvisno razvila G. G. Stokes in M. Navier v prvi polovici 19. stoletja. Enačbe Navier-Stoke razširijo Eulerjeve enačbe z opisom vpliva viskoznosti na tok tekočine. Enačbe opisujejo ohranitev gibalne tekočine (angl. conservation of momentum). Za simulacijo tekočin je, treba definirati še dve dodatni enačbi: kontinuiteto enačbo, ki opisuje ohranitev mase, ter enačbo stanja, ki opisuje ohranitev energije.

Simulacija tekočin je računsko zahteven problem, ki ga lahko razdelimo v dva podproblema. Najprej numerično aproksimiramo enačbe Navier-Stokes z metodo SPH, nato pa vizualiziramo rezultat na grafični procesni enoti (GPE). Z napredkom strojne opreme, predvsem GPE enot, so se začele razvijati tudi realno-časovne simulacije tekočin. Glavni razlogi , ki povečajo kompleksnost simulacij so procesi konvekcije, difuzija, turbulence in površinske napetosti. Animacije tekočin (angl. fluid animation) s tehnikami računalniške grafike simulirajo realistične animacije tekočin in plinov. V ospredju je predvsem vizualni aspekt, rigorozna fizikalna predstavitev gibanja tekočin je drugotnega pomena. Uporabljajo se predvsem za vizualne efekte v filmih in kot hitre animacije v računalniških igrah. Animacija tekočin je, v nasprotju s fizikalno natančno simulacijo, računsko enostavnejši problem. Glavna razlika med obema je v natančnosti rezultata. Animacije tekočin se zanašajo na aproksimacijeske rešitve Eulerjevih in Navier-Stokes enačb. Simulacija tekočin pa preučujejo fizikalno natančne pojave.

V magistrskem delu bomo najprej opisali sorodna dela iz računalniške simulacije tekočin z metodo SPH. Osredotočili se bomo na prispevek avtorjev Mueller et al. [2], ki smo ga uporabili kot osnovo za implementacijo naše rešitve. V naslednjem poglavju bomo opisali

metodo SPH in fiziko dinamike tekočin. Osredotočili se bomo na enačbe Navier-Stokes, s katerimi opisujemo gibanje šibko stisljivih tekočin (angl. weakly compressed fluids). Zatem bomo opisali lastno implementacijo algoritma simulacije tekočin z metodo SPH v 2D prostoru in predstavili rezultate na praktičnem primeru. Magistrsko delo bomo zaključili s sklepom.

2 SORODNA DELA

CFD rešuje probleme toka tekočin z uporabo naprednih numeričnih tehnik in podatkovnih struktur. Računalniška tehnologija se uporabljajo za izvedbo fizikalno natančnih simulacij tekočin in plinov z omejenimi površinami (angl. boundary conditions). Z razvojem splošne računske zmogljivosti na grafičnih procesnih enotah (angl. general purpose graphics processing unit - GPGPU), so se začele uveljavljati zahtevnejše simulacije tekočin v realnem času. Najboljše rezultate simulacij pa še vedno dobimo na visoko zmogljivih superračunalnikih. Novejše raziskave na področju CFD se osredotočajo na pojave nadzvočnih in turbulentnih tokov. Validacija programov CFD je opravljena v zračnih tunelih in v realnih testih s polnim obsegom (angl. full-scale test).

CFD spada v področje dinamike tekočin in ima dolgo zgodovino. Že leta 1822 in 1845 sta Claude Navier in George Stokes neodvisno odkrila enačbe, ki opisujejo gibanje stisljive newtonske viskozne tekočine in jih danes poznamo pod imenom enačbe Navier-Stokes. Le te predstavljajo osnovo skoraj vseh problemov na področju CFD. Z enačbami Navier-Stokes običajno opisujemo enofazni tok tekočine – npr. tok plina ali vode in ne oba hkrati.

Postopek simulacije tekočin z algoritmom CFD lahko razdelimo v več korakov. V prvem koraku predobdelamo podatke in ustrezno pripravimo simulacijsko domeno. Geometrijo in fizične omejitve problema definiramo s programom za računalniško načrtovanje (angl. computer aided design - CAD). Podatke dodatno počistimo in tako dobimo volumen oz. domeno tekočine (angl. fluid domain). Volumen tekočine nato razdelimo v diskretne celice ali mrežo. Nato definiramo fizikalne lastnosti modeliranja (npr. enačbe gibanja tekočin). Na koncu še definiramo obnašanje tekočin ob mejah simulacije. V drugem koraku izvajamo algoritem CFD, s katerim numerično izračunamo rešitve enačb Navier-Stokes. V zadnjem koraku analiziramo dobljeno rešitev, ki jo potem tudi vizualiziramo.

Enačbe Navier-Stokes opisujejo ohranitev gibalne tekočine (angl. conservation of momentum). Za uspešno simulacijo tekočin potrebujemo še enačbe, ki opišejo ohranitev mase (angl. mass conservation) in enačbo stanja, ki opisuje ohranitev energije (angl. energy conservation). V literaturi CFD je bilo predlaganih veliko numeričnih metod, ki rešujejo

zgornje enačbe. Če enačbe Navier-Stokes poenostavimo z odstranitvijo vpliva viskoznosti, dobimo Eulerjeve enačbe.

V splošnem lahko tok tekočine opišemo z dvema pristopoma. Prvi pristop temelji na Lagrangevem opisu polja toka (angl. Langrangian specification of the field) , pri katerem modeliramo tekočino z neodvisnimi gradniki tekočine – delci. V to skupino tudi uvrščamo metodo SPH in ostale brezmrežne metode. Drugi pristop, s katerim lahko opišemo tok tekočine temelji na Eulerjevem opisu polja toka (angl. Eulerian specification of the flow field). Bistvo tega pristopa je, da na tekočino gledamo v specifični točki v prostoru, skozi katero tekočina teče čez čas. Govorimo lahko tudi o Lagrangevem pristopu na podlagi delcev (angl. particle-based Langrangian approach) in Eulerjevem pristopu na podlagi mrež (angl. grid-based Eulerian approach). Langrageeov in Eulerjev opis toka tekočine sta poimenova ime po dveh velikih znanstvenikih 18. stoletja – Leonhardu Eulerju in Josephu-Louisu Lagrangeu. Slika 1 prikazuje demonstracijo Langrangeovega opisa toka tekočine in slika Slika 2 prikazuje demonstracijo Eulerjevega opisa toka tekočine.Avtorji predlagajo sisteme delcev za animacije mehkih objektov [3], [4] za animacije površin [5], za kontrolo implicitnih površin [6] in za animacije toka lave [7]. Mrežni pristop se je prav tako uveljavil za splošne simulacije tekočin [8], vode [9]–[11] mehkih objektov [12] in učinkov topljenja [13].

Na področju računalniške grafike so bile razvite specializirane tehnike za simuliranje dinamike tekočin. Leta 1983 je Reeves razvil tehniko [14] modeliranja t. i. mehkih objektov (angl. fuzzy objects) s sistemom delcev. S predlagano tehniko avtor opiše modeliranje dinamike oblakov, dima, ognja in vode. Stamova metoda stabilnih tekočin [8], ki spada v skupino pristopov na podlagi mrež, predstavlja pomemben korak k realno-časovnim simulacijam tekočin. Glavna lastnost metode je brezpogojna stabilnost, ki lahko simulira različne tekočine v 2D- in 3D prostoru. Glavna slabost metode je dejstvo, da gre samo za približek simulacije tekočin (angl. fluid-like effects) in rezultati niso fizikalno natančni. Metoda je zelo priljubljena na področju računalniške grafike, kjer so hitri in približni rezultati zaželeni. Müllerjeva metoda [2] predstavlja naslednji korak v realno-časovnih simulacijah tekočin. Avtorji razširijo tehniko Desbruna [3], ki uporabi metodo SPH za animacijo zelo deformiranih teles (angl. highly deformable bodies), za simulacijo tekočin s

prostimi površji (angl. free surface fluids). Iz enačb Navier-Stokes izpeljejo silo pritiska in viskoznosti, ter predlagajo način modeliranja površinskih napetosti. Za izboljšanje interaktivnosti algoritma, izdelajo novo gladitveno jedro. Končni rezultat predstavlja fizikalno natančna realno-časovna aplikacija.

Numerične metode, ki so značilne za mrežni pristop:

- metoda končnih razlig (angl. finite difference method - FDM),
- metoda končnih elementov (angl. finite element method - FEM),
- metoda končnega volumna (angl. finite-volume method – FVM).

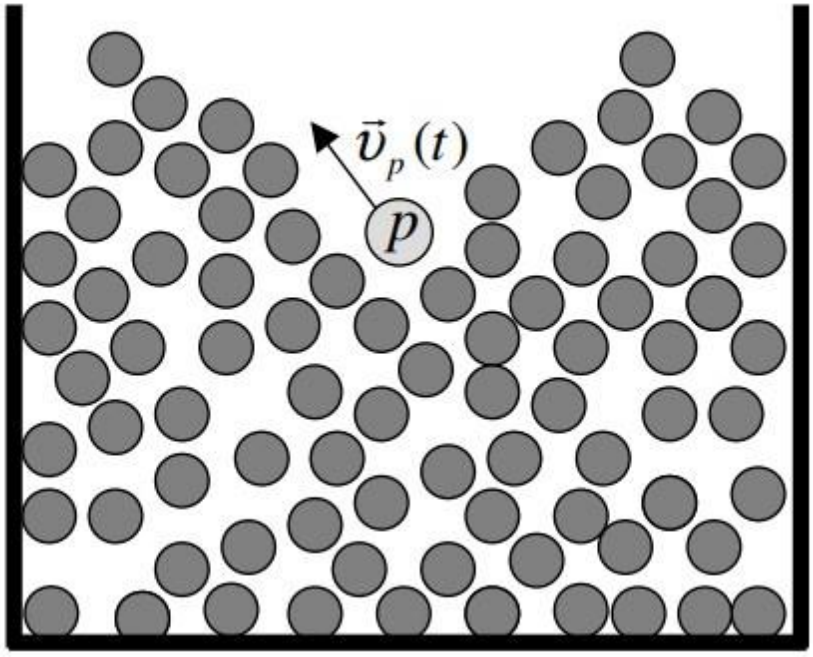
Brezmrežne metode nimajo vnaprej določenih povezav med delci v domeni simulacije (t.j. vozlišča povezana v fiksno topološko mrežo), ampak delujejo na podlagi interakcij s svojimi sosednjimi delci. Gre za skupino metod, ki spadajo v področje Langrangevega opisa polja. Delcem pripišemo lastnosti oz. veličine problema, kot so npr. masa, pritisk ali kinetična energija. Z brezmrežnimi metodami lahko simuliramo zahtevnejše probleme, ker nismo omejeni s fiksno topologijo problema. Posledično je za brezmrežne metode značilna večja računska in prostorska zahtevnost ter kompleksnejša programska izvedba.

Ena prvih razvitih brezmrežnih metod je bila metoda SPH. Ostale brezmrežne metode so:

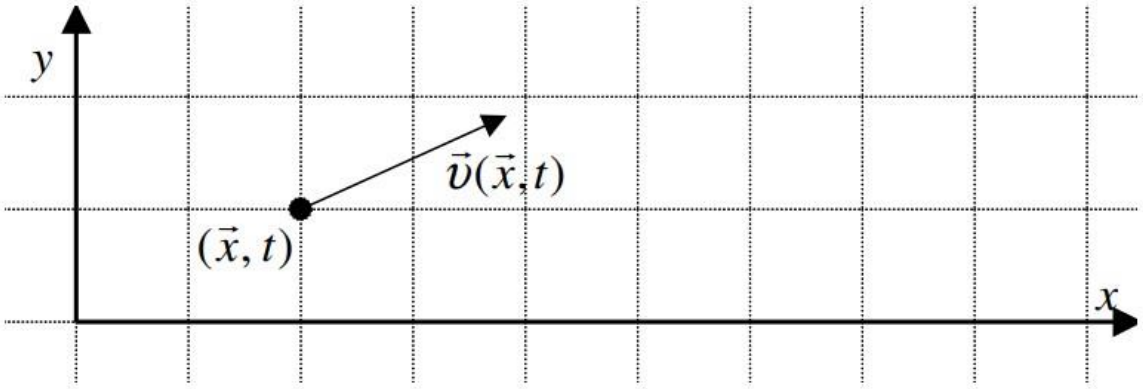
- metoda difuznih elementov (angl. Diffuse element method - DEM),
- dinamika sipajočih delcev (angl. Dissipative particle dynamics - DPD),
- Galerkinova metoda prostih elementov (angl. Element-free Galerkin method - EFGM),
- metoda delcev z reproduciranimi jedri (angl. Reproducing kernel particle method - RKPM),
- metoda končne točke (angl. Finite point method - FPM).

Obema pristopoma je skupno računanje polja hitrosti (angl. velocity field) v vzorčevalni točki. Vzorčevalne točke predstavljajo majhno območje (volumen) simulirane tekočine. V Lagrangeevem pristopu te vzorčevalne točke predstavljajo t. i. delce volumna ali mase, ki se premikajo s tokom tekočine. V Eulerjevem pristopu vzorčevalne točke predstavljajo

majhen volumen oz. mase kubične celice, ki se ne premika, ampak ostaja v fiksni topologiji prostora.



Slika 1: Ilustracija Langrangeove opisa toka z delci.



Slika 2: Ilustracija Eulerjevega opisa toka z mrežo.

V splošnem ni mogoče natančno primerjati učinkovitosti in natančnosti Lagrangeevega in Eulerjevega pristopa k opisu toka tekočine. Okvirno pa je modeliranje tekočin z Langrangevim pristopom nekoliko zahtevnejše, saj je treba določiti sosednje delce v

vsakem koraku simulacije in iz njih interpolirati vrednosti prostorskih odvodov. Medtem pa je Eulerjev pristop nekoliko učinkovitejši v računskem smislu, saj dela z nesprejemljivo topologijo polja (angl. fixed neighbourhood), ima pa Eulerjev pristop nekoli slabšo učinkovitost pri obravnavi konvekcijskih odovodov. Dodatna slabost Eulerjevega opisa toka pa je težavna predstavitev kompleksnejših fizikalnih pojavov, ki zahtevajo spremembo topološke mreže. Gre za računsko zahtevne operacije, ki posledično oslabijo natančnost simulacije. Med glavnimi slabosti spada tudi prilagodljivost Eulerjevega pristopa k dinamični geometriji, saj to zahteva visoko ločljivost celic, posledično veliko pomnilnika in računskega časa.

Tekoče simulirane z Langrangevimi pristopi in metodo SPH so stisljive zato, ker je gostota tekočine definirana s funkcijo po času. Ščasoma nastaneje oscilacije v tekočini, ki negativno vplivajo na kakovost simulacije. Nestisljivi pristopi z Eulerjevevimi metodami na drugi strani nimajo koncepta gostote tekočin. Simulacijo začnejo z mirujočo gostoto (angl. rest density) na vzorčevalni točki in izračunajo hitrost polja brez divergence v vsakem koraku simulacije, da ohranijo želeno mirujočo gostoto. Problemi, ki pa se pojavijo v teh simulacijah, so umetna viskoznost (angl. artificial viscosity), izguba mase (angl. mass loss) ali pridobitev mase (angl. mass gain).

Pri obeh pristopih torej lahko govorimo o težavah s spremembami volumna:

- v primeru stisljivih tehnik pride do sprememb volumna zaradi flaktuacij gostote in popolne ohranitve mase,
- v primeru nestisljivih tehnik se volumen spremenija zaradi spremembe mase, gostota pa je popolnoma ohranjena.

Prav tako so bile razvite tehnike, ki omogočajo uporabo Langrangevih pristopov z nestisljivimi tekočinami in Eulerjeve pristope s stisljivimi tekočinami [15]. Pri Langrangevih tehnikah oz. še posebej pri metodi SPH so bile razvite tehnike, ki imajo dovolj dobre približke z delno stisljivimi tekočinami (angl. weakly compressible fluids).

3 METODA SPH

Numerične simulacije so postale pomembno orodje k reševanju kompleksnih problemov v inženirstvu in znanosti. Z numerično simulacijo lahko fizičen problem prenesemo v diskretno matematično formulacijo in jo izvedemo z računalnikom. S tem lahko pridobimo nova znanja in se do določene mere lahko izognemo zahtevnim, dragim in včasih tudi nevarnim eksperimentom.

Metoda SPH velja za eno pomembnejših numeričnih metod za simulacijo tekočin v računalniški grafiki. Z njo lahko simuliramo kompleksne pojave s prostimi površji (angl. free-surface scenarios). Velikost simulacije je odvisna od računske zmožnosti stroja, na katerem algoritem izvajamo. Algoritem z več milijoni delcev lahko hitro simuliramo na modernih splošno računskih grafičnih procesnih enotah (angl. general purpose graphics processing unit - GPGPU). Bistvo metode je interpolacija lastnosti tekočin in aproksimacija prostorskih odvodov v enačbi stanja.

Metodo SPH sta neodvisno razvila Monagham [16] in Lucy [17] v letu 1977 za probleme na področju astrofizike, za katere so se mrežne metode izkazala za nestabilne. Kasneje se je metoda uveljavila kot pomembna metoda za simulacijo tekočin, uporablja pa se tudi na ostalih področjih fizike, balistike, vulkanologije in oceanografije [18]. Ker se je metoda izkazala za splošno uporabno, ponekod zasledimo tudi izraz algoritem dinamike zglajenih delcev (angl. smoothed particle mechanics), saj z njo lahko opišemo pojave, ki niso vezani na hidrodinamiko.

Na področju simulacije tekočin je bila metoda SPH sprva adaptirana za šibro stisljive hidrodinamike (angl. weakly compressible hydrodynamics), potem za močno stisljive (angl. strongly compressible) in nato za popolno nestisljive (angl. truly incompressible) hidrodinamike tekočin. Čeprav je metoda uporablja izrazoslovje hidrodinamike, ni vezana izključno na kapljevine, vendar se je zaradi zgodovinskih razlogov ohranilo dano ime.

Metoda SPH velja za najbolj popularno brezmrežno metodo v znanstveni literaturi in se po priljubljenosti lahko primerja z metodo MPS (angl. moving particle semi-implicit method) in metodo DPD (angl. dissipative particle dynamics). Gre za interpolacijsko metodo, ki je

aplicirana na sistem delcev. Sloni na uporabi jedrne funkcije, ki je primerna za predstavo odvodov zveznega polja v numerični obliki.

Glavne prednosti metode SPH so:

- brezmrežna topologija, ki je primerna za probleme s kompleksnimi oblikami s prostimi površji (angl. free surface), kot je npr. tok vode po hribu navzdol,
- možnost paralelne izvedbe algoritma na visoko zmogljivih računskih enotah (npr. superračunalnikih in grafično procesnih enotah z splošnim računanjem),
- možnost razširitve algoritma na probleme, ki jih definirajo polja,
- dobre ohranitve lastnosti za šibko stisljive tekočine (angl. weakly compressible SPH),
- zrela metoda, ki je dovolj natančna, stabilna in prilagodljiva za praktične inženirske aplikacije.

Glavne omejitve metode SPH so:

- računska zahtevnost,
- nenatančni rezultati ob mejah,
- napetostna nestabilnost (angl. tension instability) [19],

Metodo SPH uvrščamo v skupino brezmrežnih metod z delci (angl. mesh-free particle method). Prednosti metode SPH pri simuliranju tekočin v primerjavi z ostalimi mrežnimi metodami so:

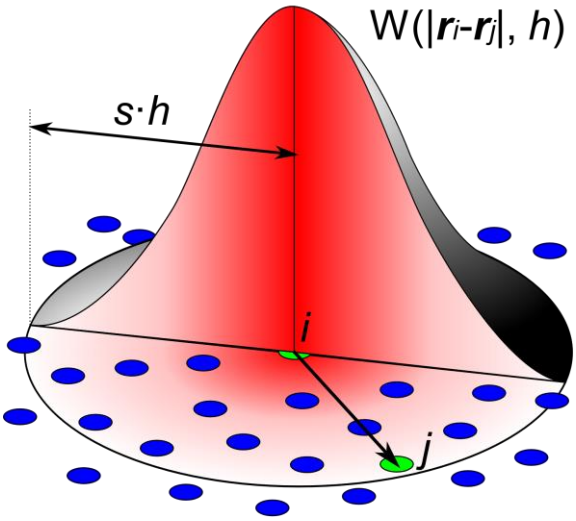
- zagotavlja ohranitev mase brez dodatnega računanja, saj delci sami predstavljajo maso,
- izračuna pritisk iz obteženih prispevkov sosednjih delcev, namesto iz računanja sistema linearnih enačb,
- direktno ustvari prosto površino za dvofazne tekočine, ker delci predstavljajo tekočino (pogosto voda), preostali prostor pa predstavlja lažjo tekočino (pogosto zrak).

Vsakemu delcu priredimo naslednje lastnosti oz. kvantitete: masa, položaj in hitrost. Tako dobimo polje lastnosti oz. kvantitet, ki ga lahko ovrednotimo kjerkoli v prostoru iz najbližjih

diskretnih elementov (delcev). Pri tem interpoliramo kvantitativne lastnosti bližnjih delcev s simetričnim glajenjem okolice. Funkciji, ki definira operacijo glajenja, z drugimi besedami pravimo jedro glajenja oz. gladitveno jedro (angl. smoothing kernel). SPH aproksimira poljubno skalarno kvantiteto A_s na lokaciji $\boldsymbol{r_i}$ kot obteženo vsoto prispevkov sosednjih delcev na naslednji način [2]:

$$A_S(\mathbf{r_i}) = \sum_j m_j \frac{A_j}{\rho_j} W(\mathbf{r_i} - \mathbf{r}_j, h) \tag{3.1}$$

Pri tem se j ponazarja indeks delca, m_j predstavlja maso delca, A_j obravnavano poljubno kvantiteto delca, ρ_j pa gostoto delca. Funkcija $W(\mathbf{r}, h)$ predstavlja gladitveno jedro z dolžino h. Uporabljeno je bilo gladitveno jedro s končno podporo, zato v našem primeru h predstavlja radij okolice. Slika 3 prikazuje bistvo metode SPH z delovanjem gladitvenega jedra pri interpolaciji kvantitete delca. Bližje, kot je delec centru jedra, večji učinek ima jedro na delec pri interpolaciji.



Slika 3: Gladitveno jedro metode SPH[1].

Masa delca in gostote je podana v enačbi (3.1), ker vsak delec *i* zavzema določen volumen $V_i = \frac{m_i}{\rho_i}$. Masa m_i je konstantna skozi simulacijo in enaka za vse delce. Gostota ρ_i se

[1] Vir slike: https://en.wikipedia.org/wiki/Smoothed-particle_hydrodynamics#/media/File:SPHInterpolationColorsVerbose.svg

spreminja in jo je treba izračunati na vsakem časovnem (integracijskem) koraku. Skozi substitucijo v enačbi (3.1) dobimo enačbo za gostoto na lokaciji **r** [2]:

$$\rho_S(\mathbf{r}) = \sum_j m_j \frac{\rho_j}{\rho_j} W(\mathbf{r_i} - \mathbf{r}_j, h) = \sum_j m_j W(\mathbf{r_i} - \mathbf{r}_j, h) \tag{3.2}$$

V večini simulacij tekočin se pojavljajo prostorski oz. materialni odvodi kvantitetnega polja, zato jih je treba izpeljati in definirati tudi pri metodi SPH. Prostorski odvodi pri metodi vplivajo samo na gladitveno jedro. Gradient *A* je definiran kot:

$$\nabla A_S(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j} \nabla W(\mathbf{r_i} - \mathbf{r}_j, h) \tag{3.3}$$

Laplaceov operator *A* je definiran kot [2]:

$$\nabla^2 A_S(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j} \nabla^2 W(\mathbf{r_i} - \mathbf{r}_j, h) \tag{3.4}$$

Treba je poudariti, da metoda SPH prinaša nekaj slabosti. Ko uporabimo metodo SPH za izpeljavo enačb za gibanje tekočin z delci, potem enačbe ne zagotavljajo doslednega upoštevanja nekaterih fizikalnih zakonov, kot so zakon o simetriji sil (t. j. tretji Newtonow zakon) in zakon o ohranitvi gibalne količine.

3.1 Teoretična podlaga

Osnovne enačbe, ki jih je treba definirati za simulacijo tekočine, so:

- enačbe Navier-Stokes, ki opisujejo ohranitev gibalne tekočine (angl. conservation of mementum),
- enačbo kontinuitete, ki opisuje ohranitev mase (angl. conservation of mass) in
- enačbo stanja, ki opisuje ohranitev energije (angl. conservation of energy).

V Eulerjevi formulaciji z mrežnim pristopom, izotermalne tekočine opišemo s poljem hitrosti **v**, s poljem gostote ρ in s poljem pritiska *p*. Evolucija teh veličin skozi čas opisujeta dve enačbi. Prva enačba zagotavlja ohranitev mase [2]:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \tag{3.5}$$

Enačba Navier-Stoke zagotavlja ohranitev gibalne količine [2]:

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v}\right) = -\nabla p + \rho \mathbf{g} + \mu \nabla^2 \mathbf{v}, \tag{3.6}$$

kjer je $\mathbf{g}$ gravitacijska oz. zunanja sila in μ je viskoznost tekočine. Veliko oblik enačbe Navier-Stokes se pojavlja v literaturi. Enačba (3.6) predstavlja poenostavljeno obliko za nestisljive tekočine.

Uporaba delcev namesto stacionarne mreže poenostavi zgornji dve enačbi. Ker je število delcev konstantno in vsak delec ima enako (nespremenljivo) maso, je tako zagotovljena ohranitev mase in lahko izpustimo enačbo (3.5) iz simulacije. Druga poenostavitev je v enačbi (3.6). Izraz $\partial \mathbf{v}/\partial t + \mathbf{v} \cdot \nabla \mathbf{v}$ lahko zapišemo s snovnim oz. materialnim odvodom (angl. material derivative) $D\mathbf{v}/Dt$. Ker se delci gibljejo s tokom, je materialni odvod hitrostnega polja preprosto časovni odvod hitrosti delca, kar pomeni, da izraz $\mathbf{v} \cdot \nabla \mathbf{v}$ ni potreben za sistem delcev. Na desni strani enačbe (3.6) so tri polja sil, ki modelirajo:

- pritisk $-\nabla p$,
- gravitacijo $\rho \mathbf{g}$ in
- viskoznost tekočine $\mu \nabla^2 \mathbf{v}$.

Njihov seštevek nam da skupno silo, ki deluje na tekočino v danem času in prostoru [2]:

$$\mathbf{F_S} = -\nabla p + \rho \mathbf{g} + \mu \nabla^2 \mathbf{v} \tag{3.7}$$

Sprememba sil v (3.7) določa pospešek delcev in posledično spremembo hitrosti obravnavanih delcev v tekočini. Za pospešek delca *i* dobimo enačbo:

$$\mathbf{a}_i = \frac{d\mathbf{v_i}}{dt} = \frac{\mathbf{F_i}}{\rho_i} \tag{3.8}$$

Kjer $\mathbf{v_i}$ predstavlja hitrost delca *i*, $\mathbf{F_i}$ vektor sil in ρ_i gostoto ovrednoteno za lokacijo delca *i*.

Zgornje enačbe ne definirajo interakcije tekočin s trdnimi snovmi in z (navideznimi) mejami simulacije. Posebna skrbnost je potrebna pri obravnavi mej, da ne pride do napak v simulaciji zaradi neustreznih interakcij tekočin z okolico. V večini implementacij SPH so trdne snovi vzorčene z implicitnimi matematičnimi funkcijami ali z delci, katerih sila deluje pravokotno iz površja objekta (v nasprotni smeri približujočega se delca). Nekateri avtorji

[18] predlagajo izračun sile objekta na podlagi razdalje z delcem. S tem lahko simuliramo Lennard-Jonesove sile, ki polinomsko padajo z razdaljo med delcem in objektom. Ob tem pa lahko pride do velikih variacij pritiska v tekočini, ki so značilne predvsem za šibko stisljive tekočine. Posledično je treba znižati časovni korak simulacije, ki negativno vpliva na interaktivnost končne aplikacije. Becker predlaga korekcijsko metodo [20] z neposrednim apliciranjem sile.

3.2 Definiranje delcev in gladitvenega jedra okolice

Metoda SPH aproksimira tekočino z velikim številom delcev. Vsakemu delcu pripišemo lastnosti, ki se premikajo z njim po prostoru. Več kot je delcev, bolj je simulacija fizikalno natančna, posledično pa tudi računsko zahtevnejša. Uporabili bomo gladitvnega jedra, ki jih je predlagal Müller [21]. Na hitrost, stabilnost in natančnost metode SPH vpliva izbira gladitvenega jedra. Pogoj dobrega gladitvenega jedra je njegov red napake. Želimo definirati gladitveno jedro vsaj drugega reda. Gladitveno jedro ima interpolacijsko napako drugega reda, če je funkcija jedra:

- soda:

$$W(\mathbf{r}, h) = W(-\mathbf{r}, h)) \tag{3.9}$$

- normalizirana:

$$\int W(\mathbf{r}) d\mathbf{r} = 1 \tag{3.10}$$

Dodatno so jedra katerih odvod se približuje ničli ob mejah stabilnejša. Müller je zasnoval splošno gladitveno jedro za 3D prostor, ki je primerno za večino splošnih simulacij tekočine. V nadaljevanju bomo predstavili tri različna jedra v 2D in 3D izvedbi, njihove gradiente in Laplaceove operatorje. Naslednja enačba predstavlja Müllerjevo jedro v 2D in 3D izvedbi [2]:

$$W_{poly6,2D}(\mathbf{r}, h) = \frac{4}{\pi h^8} \begin{cases} (h^2 - \|\mathbf{r}\|^2)^3, & 0 \leq \|\mathbf{r}\| \leq h \\ 0, & \|\mathbf{r}\| > h \end{cases} \tag{3.11}$$

$$W_{poly6,3D}(\mathbf{r}, h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - \|\mathbf{r}\|^2)^3, & 0 \leq \|\mathbf{r}\| \leq h \\ 0, & \|\mathbf{r}\| > h \end{cases} \tag{3.12}$$

Gradient zgornjega jedra za 2D in 3D prostor je definiran kot:

$$\nabla W_{poly6,2D}(\mathbf{r},h) = -\frac{24}{\pi h^8}\begin{cases}(h^2-\|\mathbf{r}\|^2)^2\mathbf{r}, & 0 \le \|\mathbf{r}\| \le h \\ 0, & \|\mathbf{r}\| > h\end{cases} \tag{3.13}$$

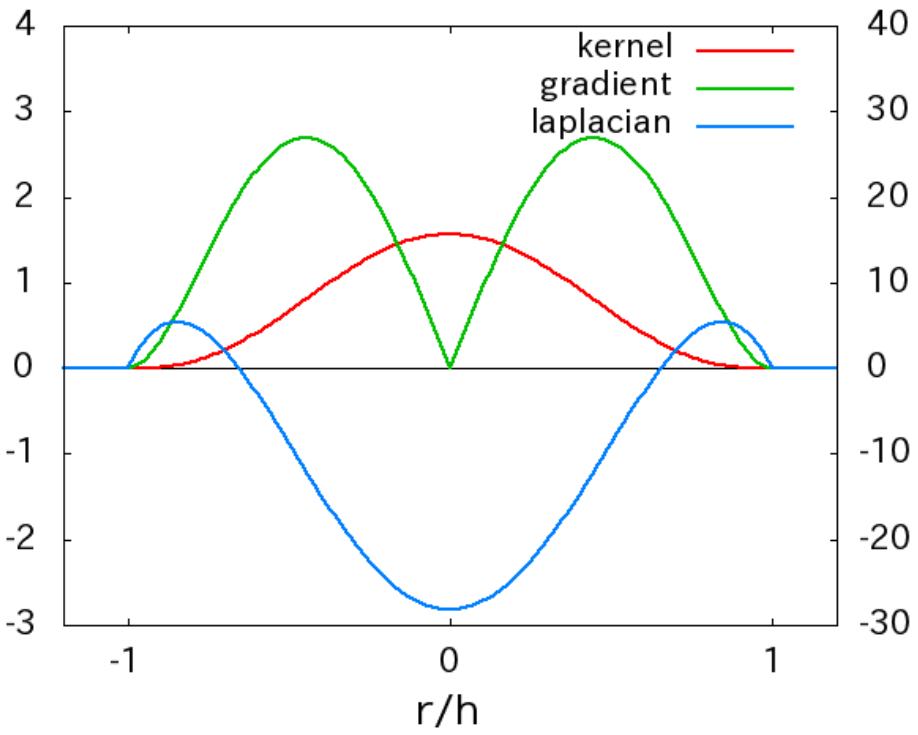
$$\nabla W_{poly6,3D}(\mathbf{r},h) = -\frac{945}{32\pi h^9}\begin{cases}(h^2-\|\mathbf{r}\|^2)^2\mathbf{r}, & 0 \le \|\mathbf{r}\| \le h \\ 0, & \|\mathbf{r}\| > h\end{cases} \tag{3.14}$$

Laplaceov operator zgornjega jedra za 2D in 3D prostor je definiran kot:

$$\nabla^2 W_{poly6,2D}(\mathbf{r},h) = -\frac{24}{\pi h^8}\begin{cases}3(h^2-\|\mathbf{r}\|^2)^2 - 4\|\mathbf{r}\|^2(h^2-\|\mathbf{r}\|^2)^2, & 0 \le \|\mathbf{r}\| \le h \\ 0, & \|\mathbf{r}\| > h\end{cases} \tag{3.15}$$

$$\nabla^2 W_{poly6,3D}(\mathbf{r},h) = -\frac{945}{32\pi h^9}\begin{cases}3(h^2-\|\mathbf{r}\|^2)^2 - 4\|\mathbf{r}\|^2(h^2-\|\mathbf{r}\|^2)^2, & 0 \le \|\mathbf{r}\| \le h \\ 0, & \|\mathbf{r}\| > h\end{cases} \tag{3.16}$$

Slika 4 prikazuje normalizirano funkcijo jedra poly6, njen gradient in Laplaceov operator.



Slika 4: Grafi funkcije, gradienta in Laplaceovega operatorja gladitvenega jedra poly6[2].

[2] Vir slike: http://www.slis.tsukuba.ac.jp/~fujisawa.makoto.fu/cgi-bin/wiki/index.php?plugin=attach&refer=SPH%CB%A1%A4%CE%BD%C5%A4%DF%B4%D8%BF%F4&openfile=kernel_poly6.png

Pomembna lastnost tega jedra je, da se **r** pojavi v kvadratni obliki. To pomeni, da lahko izračunamo razdaljo, ne da bi bilo treba izračunati kvadratni koren. Če to jedro uporabimo za računanje sile pritiska, potem se delci zbirajo v gručah pod velikim pritiskom. Ko delci pridejo zelo blizu skupaj, se gradient jedra približa ničli v centru in odbojna sila izgine oz. izgubi svoj namen. Za rešitev tega problema Desbrun predlaga novo jedro imenovano spičasto (angl. *spiky*) brez ničelnega gradienta v sredini [3]:

$$W_{spiky,2D}(\mathbf{r},h)=\frac{10}{\pi h^5}\begin{cases}(h-\|\mathbf{r}\|)^3, & 0\le\|\mathbf{r}\|\le h\\ 0, & \|\mathbf{r}\|>h\end{cases} \tag{3.17}$$

$$W_{spiky,3D}(\mathbf{r},h)=\frac{15}{\pi h^6}\begin{cases}(h-\|\mathbf{r}\|)^3, & 0\le\|\mathbf{r}\|\le h\\ 0, & \|\mathbf{r}\|>h\end{cases} \tag{3.18}$$

Gradient zgornjega jedra je za 2D in 3D prostor definiran kot:

$$\nabla W_{spiky,2D}(\mathbf{r},h)=-\frac{30}{\pi h^5}\begin{cases}(h-\|\mathbf{r}\|)^2\dfrac{\mathbf{r}}{\|\mathbf{r}\|}, & 0\le\|\mathbf{r}\|\le h\\ 0, & \|\mathbf{r}\|>h\end{cases} \tag{3.19}$$

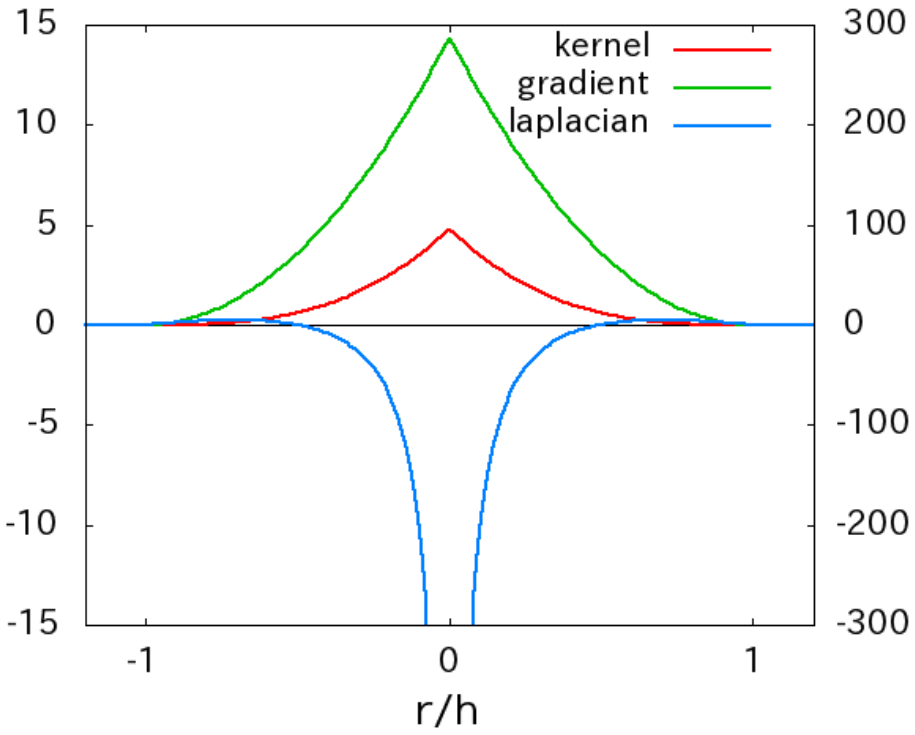
$$\nabla W_{spiky,3D}(\mathbf{r},h)=-\frac{45}{\pi h^6}\begin{cases}(h-\|\mathbf{r}\|)^2\dfrac{\mathbf{r}}{\|\mathbf{r}\|}, & 0\le\|\mathbf{r}\|\le h\\ 0, & \|\mathbf{r}\|>h\end{cases} \tag{3.20}$$

Laplaceov operator zgornjega jedra je za 2D in 3D prostor definiran kot:

$$\nabla^2 W_{spiky,2D}(\mathbf{r},h)$$
$$=-\frac{60}{\pi h^5}\begin{cases}\dfrac{1}{\|\mathbf{r}\|}(h-\|\mathbf{r}\|)^2-(h-\|\mathbf{r}\|), & 0\le\|\mathbf{r}\|\le h\\ 0, & \|\mathbf{r}\|>h\end{cases} \tag{3.21}$$

$$\nabla^2 W_{spiky,3D}(\mathbf{r},h)$$
$$=-\frac{90}{\pi h^6}\begin{cases}\dfrac{1}{\|\mathbf{r}\|}(h-\|\mathbf{r}\|)^2-(h-\|\mathbf{r}\|), & 0\le\|\mathbf{r}\|\le h\\ 0, & \|\mathbf{r}\|>h\end{cases} \tag{3.22}$$

Slika 5 prikazuje upodobljeno funkcijo jedra spiky, njen gradient in Laplaceov operator.



Slika 5: Grafi funkcije, gradienta in Laplaceovega operatorja gladitvenega jedra spiky[3].

Dobra lastnost tega jedra sta prvi odvor (gradient) in drugi odvod (Laplaceov operator), ki se oba približata ničli na meji področja.

Viskoznost je fenomen, ki je merilo odpora tekočine proti strižni napetosti. Različne tekočine imajo različne vplive viskoznosti. Zaradi manjše viskoznosti teče voda hitreje od medu. Zaradi velike viskoznosti med le s težavo mešamo. Viskoznost medu se zmanjša, če ga segrejemo. V tabeli Tabela 1 so podane viskoznosti in gostote nekateri snovi pri temperaturi 20 °C in tlaku 1 bar.

[3] http://www.slis.tsukuba.ac.jp/~fujisawa.makoto.fu/cgi-bin/wiki/index.php?plugin=attach&refer=SPH%CB%A1%A4%CE%BD%C5%A4%DF%B4%D8%BF%F4&openfile=kernel_spiky.png

Tabela 1: Viskoznosti in gostote nekaterih snovi.

snov	**viskoznost η [10^{-3} Pa s]**	**gostota ρ [kg dm^{-3}]**
voda	1,0	1,0
zrak	0,018	0,0012
vodik	0,0090	0,000082
olivno olje	81	0,92
motorno olje SAE 10	65	0,86
motorno olje SAE 40	320	0,9
med	2000 - 10000	1,4
Kri	3 – 4	1,04

Viskoznost se kaže kot notranje trenje med plastmi gibajoče tekočine. S tem zmanjša kinetično energijo tekočine, ki se pretvori v toploto. Zato ima viskoznost efekt glajenja na polje hitrosti. Če uporabimo jedro poly6 definirano v (3.11) in (3.12), potem rezultirajoče sile viskoznosti ne zagotavljajo lastnosti glajenja. Ko se delca približata drug drugemu, lahko Laplaceov operator glajenega polja hitrosti daje negativne rezultate sile, ki povečajo relativne hitrosti. Ta artefakt se še posebej izrazi v grobo vzorčenih poljih hitrosti. V realnočasovnih aplikacijah, kjer je število delcev relativno malo, lahko predstavljen efekt povzroča probleme s stabilnostjo. Za računanje sile viskoznost, Müller predstavi naslednje jedro [2]:

$$W_{visc,2D}(\mathbf{r}, h) = \frac{10}{3\pi h^2}\begin{cases} -\frac{\|\mathbf{r}\|^3}{2h^3} + \frac{\|\mathbf{r}\|^2}{h^2} + \frac{h}{2\|\mathbf{r}\|} - 1, & 0 \le \|\mathbf{r}\| \le h \\ 0, & \|\mathbf{r}\| > h \end{cases} \tag{3.23}$$

$$W_{visc,3D}(\mathbf{r},h)$$
$$= \frac{15}{2\pi h^3}\begin{cases} -\frac{\|\mathbf{r}\|^3}{2h^3} + \frac{\|\mathbf{r}\|^2}{h^2} + \frac{h}{2\|\mathbf{r}\|} - 1, & 0 \le \|\mathbf{r}\| \le h \\ 0, & \|\mathbf{r}\| > h \end{cases} \quad (3.24)$$

Gradient zgornjega jedra v 2D in 3D prostoru je definiran kot:

$$\nabla W_{visc,2D}(\mathbf{r},h)$$
$$= \frac{10}{3\pi h^4}\begin{cases} \left(-\frac{3r}{2h} + 2 - \frac{h}{2\|\mathbf{r}\|^3}\right)\mathbf{r}, & 0 \le \|\mathbf{r}\| \le h \\ 0, & \|\mathbf{r}\| > h \end{cases} \quad (3.25)$$

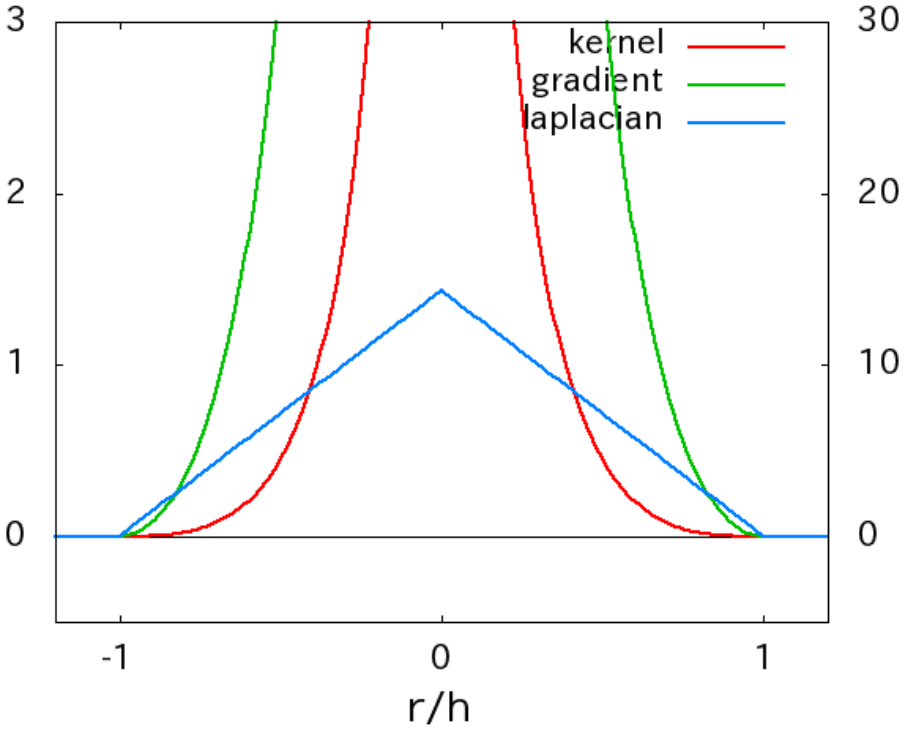
$$\nabla W_{visc,3D}(\mathbf{r},h)$$
$$= \frac{15}{2\pi h^5}\begin{cases} \left(-\frac{3r}{2h} + 2 - \frac{h}{2\|\mathbf{r}\|^3}\right)\mathbf{r}, & 0 \le \|\mathbf{r}\| \le h \\ 0, & \|\mathbf{r}\| > h \end{cases} \quad (3.26)$$

Laplaceov operator zgornjega jedra v 2D in 3D prostoru je definiran kot:

$$\nabla^2 W_{viskoznost,2D}(\mathbf{r},h) = \frac{20}{3\pi h^5}\begin{cases} (h - \|\mathbf{r}\|), & 0 \le \|\mathbf{r}\| \le h \\ 0, & \|\mathbf{r}\| > h \end{cases} \quad (3.27)$$

$$\nabla^2 W_{viskoznost,3D}(\mathbf{r},h) = \frac{45}{\pi h^6}\begin{cases} (h - \|\mathbf{r}\|), & 0 \le \|\mathbf{r}\| \le h \\ 0, & \|\mathbf{r}\| > h \end{cases} \quad (3.28)$$

Na sliki Slika 6 vidimo funkcijo jedra viskoznosti, njegovega gradienta in Laplaceovega operatorja.



Slika 6: Grafi funkcije, gradienta in Laplaceovega operatorja gladitvenega jedro za viskoznost

V enačbi (3.26) je Laplaceov operator povsod pozitiven in ima naslednje lastnosti:

$$\begin{aligned} \nabla^2 W(\mathbf{r}, h) &= \frac{45}{\pi h^6}(h - r) \\ W(|\mathbf{r}| = h, h) &= 0 \\ \nabla W(|\mathbf{r}| = h, h) &= \mathbf{0} \end{aligned} \tag{3.29}$$

Uporaba predlaganega jedra za računanje sile viskoznosti poveča stabilnost simulacije in odstrani potrebo po dodatnem dušenju (angl. damping).

3.3 Gostota, viskoznost in pritisk

Aplikacija enačbe SPH na člen pritiska $-\nabla p$ da naslednjo enačbo [2]:

$$\mathbf{F}_i^{pritisk} = -\nabla p(\mathbf{r}_i) = -\sum_j m_j \frac{p_j}{p_j} \nabla W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{3.30}$$

Slabost enačbe (3.30) je nesimetričnost pritiska, kar lahko opazimo pri interakciji dveh delcev. Ker je gradient jedra enak nič v svojem centru, delec *i* preprosto uporabi pritisk delca *j* in obratno. Pritiski na lokaciji dveh delcev v splošnem niso enaki, zato sile ne bodo

simetrične. V literaturi je bilo predlaganih več rešitev problema simetričnosti sile, Müller pa predlaga naslednjo enačbo [2]:

$$\mathbf{F}_i^{pritisk} = -\sum_j m_j \frac{p_j + p_i}{2p_j} \nabla W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{3.31}$$

Izračunana sila pritiska je simetrična, saj uporabi aritmetično sredino pritiska delcev v interakciji. Ker delci nosijo informacijo mase, pozicije in hitrosti, je najprej treba izračunati pritisk na lokaciji delca. To naredimo v dveh korakih. Z enačbo (3.2) izračunamo gostoto na lokaciji delca. Pritisk lahko izračunamo iz enačbe stanja za idealni plin (angl. ideal gas state equation):

$$p = k\rho \tag{3.32}$$

Kjer je *k* plinska konstanta, ki je odvisna od temperature in ρ predstavlja gostoto tekočine. V naši simulaciji bo uporabljena spremenjena oblika enačbe (3.32) predlagana s strani Desbruna [3]:

$$p = k(\rho - \rho_0) \tag{3.33}$$

Kjer je ρ_0 mirujoča gostota (angl. rest density) tekočine. Ker so sile pritiska odvisne od gradienta polja pritiska, odmik matematično ne spreminja sil pritiska. Vpliva pa na gradient polja, ki je zglajen s SPH in naredi simulacijo numerično stabilnejšo.

Posledica aplikacije metode SPH na člen viskoznosti $\mu\nabla^2\mathbf{v}$ so asimetrične sile:

$$\mathbf{F}_i^{viskoznost} = \mu\nabla^2\mathbf{v}(\mathbf{r}_a) = \mu\sum_j m_j \frac{\mathbf{v}_j}{\rho_j} \nabla^2 W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{3.34}$$

To pa zato, ker se polje hitrost spreminja od delca do delca. Ker so sile viskoznosti odvisne samo od sprememb v hitrosti in ne od absolutnih hitrosti, lahko silo viskoznosti transformiramo v simetrično z razliko hitrosti:

$$\mathbf{F}_i^{viskoznost} = \mu\sum_j m_j \frac{\mathbf{v_j} - \mathbf{v_i}}{\rho_j} \nabla^2 W(\mathbf{r_i} - \mathbf{r_j}, h) \tag{3.35}$$

Enačbo (3.35) lahko interpretiramo tako, da pogledamo na sosede delca *i* z njegovega premikajočega se referenčnega okvirja. Potem je delec i pospešen v smeri relativne hitrosti njegovega okolja.

Molekule v tekočini so podvržene privlačnim silam sosednjih molekul. Tem silam pravimo površinske napetosti. Sile površinskih napetosti niso zajete v enačbi (3.6), zato Müller predlaga uporabo Morrisovega modela [22]. Znotraj tekočin so medmulekularne sile enakomerne v vseh smereh in se med seboj izničijo. Pri prostih površjih (angl. free surfaces) pa sile na molekule niso več enakomerne. Seštevek sil (angl. net force) deluje iz smeri normale prostega površja proti tekočini. Sile prav tako zmanjšajo ukrivljenost površja. Večja, kot je ukrivljenost, večja je končna sila. Na površinsko napetost prav tako vpliva koeficient trenja σ (angl. friction coefficient), ki ga določata dve tekočini, ki tvorita prosto površje.

Površje tekočine lahko določimo z novo lastnostjo polja, ki jo v literaturi zasledimo pod imenom barvno polje (angl. color field). Vrednost barvnega polja določimo 1 na lokacijah delcev in 0 povsod drugje. Zglajeno barvno polje opisuje enačba (3.36).

$$\mathrm{c}_s = \sum_j m_j \frac{1}{\rho_j} W(\mathbf{r_i} - \mathbf{r_j}, h) \tag{3.36}$$

Gradient glajenega barvnega polja v (3.37) določa normalo površja, ki kaže proti tekočini in divergenca polja **n** v (3.38) določa ukrivljenost površja.

$$\mathbf{n} = \nabla \mathrm{c}_s \tag{3.37}$$

$$\kappa = \frac{\nabla^2 \mathrm{c}_s}{|\mathbf{n}|} \tag{3.38}$$

Ko združimo zgornje enačbe, lahko definiramo oprijem površja (angl. surface traction) [2]:

$$\mathbf{t}^{površje} = \sigma\kappa \frac{\mathbf{n}}{|\mathbf{n}|} \tag{3.39}$$

Enačbo (3.39) pomnožimo z normaliziranim skalarnim poljem $\delta_s = |\mathbf{n}|$, saj je le ta definiran samo blizu površja tekočine. Tako lahko definiramo silo površja v (3.40).

$$\mathbf{F}^{površje} = \sigma\kappa\mathbf{n} = -\sigma\nabla^2 c_s \frac{\mathbf{n}}{|\mathbf{n}|} \tag{3.40}$$

3.4 Časovna integracija

Bistvo fizikalnih simulacij je napovedovanje naslednjega stanja sistema na podlagi trenutnega stanja ob aplikaciji fizikalnih zakonov. Napovedi so relativno preproste, kjer napovemonaslednjo pozicijo delca, ki se premika z znano hitrostjo hitrostjo v določeni smeri. Te napovedi izračunamo z matematično tehniko, ki se imenuje numerična integracija. Gre za tehniko, ki se uporablja za probleme, katerih rešitev ni mogoče enostavno izračunati z analitičnimi metodami in simbolnim računanjem (angl. symbolic computation). Tehnika je zato primerna za mnoge inženirske probleme, kjer se zadovoljimo s približnimi rešitvami problema v relativno hitrem času.

Dolžina časovnega koraka Δt je zelo pomemben del numerične simulacije. Izbira časovnega koraka znatno vpliva na pravilno delovanje simulacije. Nepravilna izbira časovnega koraka lahko vodi v nestabilnosti in napačne rezultate. Optimalna izbira časovnega koraka je pomembna zato, da lahko dosežemo maksimalno hitrost in ohranimo fizikalno natančno simulacijo.

Pomembnejše metode za časovno integracijo fizikalnih simulacij so:

- Delno-implicitna Eulerjeva metoda (angl. Semi-implicit Euler method),
- Verletova integracija s hitrostjo (angl. Velocity Verlet),
- Metoda žabjega koraka (angl. Leapfrog method).

3.4.1 Delno-implicitna Eulerjeva metoda

Eulerjeva metoda (angl. Euler method) je najbolj znana in najpreprostejša integracijska metoda. Poznamo jo tudi pod imenom eksplicitna ali napredujoča Eulerjeva metoda (angl. explicit or forward Euler method). Bistvo metode je, da v istem časovnem koraku neodvisno izračunamo položaj (3.41) in hitrost (3.42).

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \mathbf{v}(t)\Delta t \tag{3.41}$$

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \mathbf{a}(t)\Delta t \tag{3.42}$$

Obstaja tudi implicitna varianta Eulerjeve metode, ki rešuje hibe eksplicitne metode (npr. premajhen časovni korak, ki je numerično nestabilen) in velja za računsko zahtevnejšo metodo. Delno-implicitna Eulerjeva metoda, ki je izpeljana iz implicitne Eulerjeve metode, pa v istem koraku uporabi izračunano hitrost, za določanje naslednjega položaja (3.43).

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \mathbf{v}(t + \Delta t)\Delta t \tag{3.43}$$

3.4.2 Verletova integracija s hitrostjo

Verletova integracija je numerična metoda, ki se pogosto uporablja za integracijo Newtonovih enačb gibanja, za računanje trajektorij delcev v molekularni dinamiki in v računalniški grafiki. Prvi jo je uporabil francoski matematik Delambre konec 18. stoletja, potem pa je bila večkrat na novo odkrita. Nazadnje v 60-ih letih 20. stoletja, ko jo je odkril Loup Verlet in jo uporabil za probleme molekularne dinamike [23]. Včasih zasledimo tudi ime Störmerjeva metoda (oz. Störmer-Verlet), ki nosi ime po fiziku Stormerju, ki je metodo ponovno odkril na začetku 20. stoletja in jo uporabil za računanje orbite Halleyevega kometa [24]. Verletova integracija zagotavlja numerično stabilnost, časovno reverzibilnost in ohranitev simplektične oblike v prostoru, obnem pa ni računsko zahtevnejša od enostavnejše Eulerjeve metode.

Napredna izvedba Verletove integracijske metode se imenuje Verletova integracija s hitrostjo (angl. velocity Verlet), ki reši problem prvega časovnega koraka v osnovni Verletovi metodi [25]. Enačbi (3.44) in (3.45) opišeta hitrost in pozicijo napredne Verletovo metode.

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)\Delta t^2 \tag{3.44}$$

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \frac{\mathbf{a}(t) + \mathbf{a}(t + \Delta t)}{2}\Delta t \tag{3.45}$$

Verletova integracija s hitrostjo je poseben primer metode Newmark-beta s parametri β=0 in γ=1/2.

3.4.3 Metoda žabjega koraka

Integracija po metodi žabjega koraka (angl. Leapfrog method) oz. preskočna integracija je integracijska metoda drugega reda natančnosti po času in prostoru. Integracija po Eulerjevi

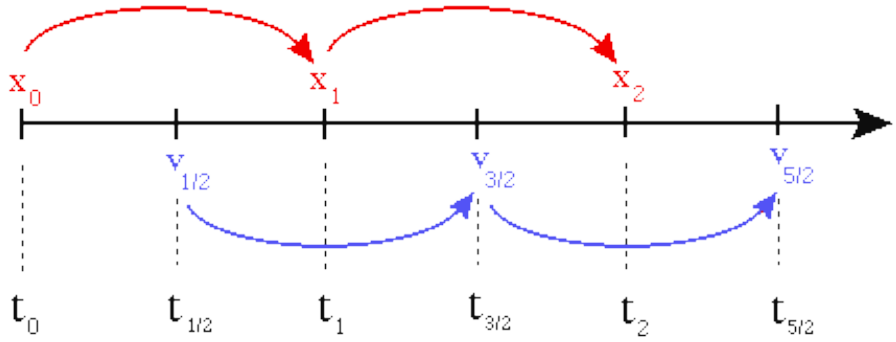
metodi je metoda prvega reda. Obe metodi zahtevata enako število evaluacij funkcije na korak simulacije. Metoda žabjega koraka je stabilna za oscilirajoče gibanje, integracija po Eulerjevi metodi pa ne. V tej magistrski nalogi bomo najprej preučili in implementirali Eulerjevo metodo, ki je preprostejša za izvedbo, nato pa še metodo žabjega koraka, ki je numerično stabilnejša.

Metoda je dobila ime po dejstvu, da računanje hitrosti preskakuje računanje položaja delcev za polovico časovnega koraka. Pri Eulerjevi in Verletovi integracijski metodi pa hitrost in položaj lahko izračunamo ob istem časovnem koraku.

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)\Delta t^2 \tag{3.46}$$

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t) + \frac{\mathbf{a}(t) + \mathbf{a}(t+\Delta t)}{2}\Delta t \tag{3.47}$$

Slika 7 prikazuje osnovno idejo metode žabjega preskoka.

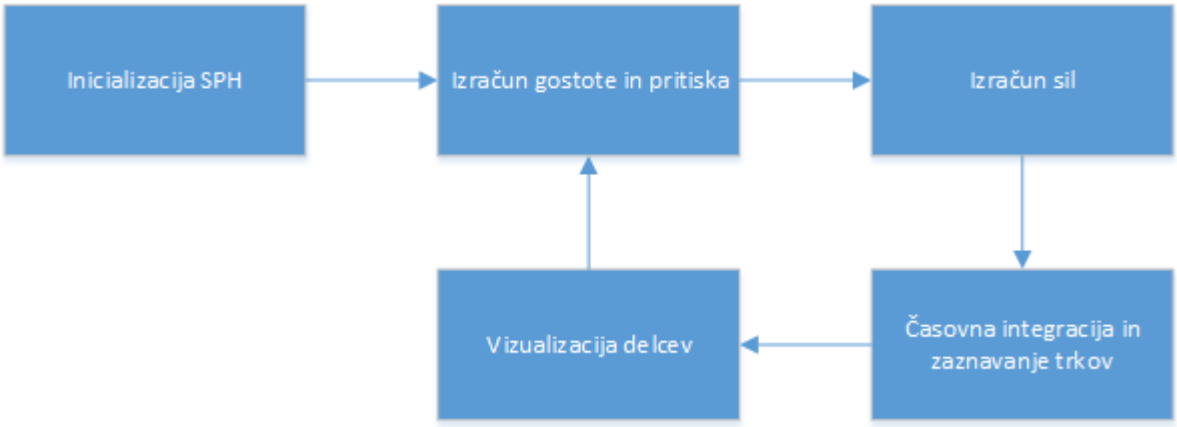


Slika 7: Ilustracija metode žabjega preskoka[4].

[4] Vir slike: http://cvarin.github.io/CSci-Survival-Guide/images/leapfrog.gif

4 IMPLEMENTACIJA

V praktičnem delu magistrske naloge bomo implementirali simulacijo tekočin v 2D prostoru z metodo SPH. Pri implementaciji se bomo zgledovali po avtorjih v članku [2]. Aplikacijo bomo implementirali s programskim jeziku C++, pri čemer bomo prototip algoritma najprej implementirali v programskem jeziku Pyhon. Animacijo bomo izrisali z grafično knjižnico OpenGL. Slika 8 prikazuje osnovni potek simulacije tekočine v našem programu. V tem poglavju bomo podrobneje predstavili vsak korak izvedenega simulatorja. Ker je simulacija tekočin široko področje, se bomo v praktični izvedbi večkrat omejili. Glavni namen programa je demonstracija metode SPH in njenih pomembnejših konceptov na praktičnem primeru simulacije toka vode. Računsko optimalna izvedba algoritma ni cilj te demonstracije, zato koda ni primerna za industrijske aplikacije. Na mestih v programu, za katera vemo, da obstajajo optimalnejše rešitve, na to opozorimo in jih na kratko predstavimo.



Slika 8: Potek aglortma za simulacijo po metodi SPH.

Na trgu je na voljo več komercialnih, prosto dostopnih in akademskih rešitev za animacijo tekočin [26]–[28]. Najnaprednejše izvedbe za računanje uporabljajo moderne grafične procesne enote, ki ponujajo GPGPU. Dane enote ponujajo enormno računsko moč, ki jo pisci aplikacij s pridom uporabijo, ter tako zagotovijo interaktivnost simulacij in njihovo fizikalno natančnost. Slabost opisanega pristopa je kompleksnejša programska rešitev.

4.1 Iskanje sosedov

Ena najpogostejših operacij v simulaciji z delci je iskanje sosedov oz. najbližjih delcev v dani okolici. Na področju računalniške geometrije je bilo razvitih več algoritmov iskanja bližnjih sosedov (angl. nearest neighbor search – NNS), ki se uporabljajo predvsem pri iskanju trkov (angl. collision detection) [29], [30]. Ovrednotenje enačb SPH se izvede za vse delce simulacije. Vsakemu delcu je treba določiti sosede s pomočjo katerih se aproksimirajo obravnavane veličine simulacije. Naivna implementacija iskanja sosedov za *n* delcev se izvede s kvadratno časovno zahtevnostjo - $O(n^2)$. Z večanjem števila delcev se zmanjša interaktivnost aplikacije, zato bomo implementirali hitro iskanje sosedov po principu prostorske zgoščevalne funkcije. Prostor razdelimo na celice fiksnih velikosti in izkoristimo dejstvo, da so gladitvena jedra omejena (z vrednostjo parametra *h*), zato omejimo iskanje sosedov samo v bližnjih celicah. Časovna kompleksnost se zmanjša na O(n*m), kjer je *m* povprečno število najdenih delcev.

Metoda prostorske zgoščevalne funkcije [31] je hiter algoritem NNS, ki opravi iskanje v konstantnem času O(1). Metoda za vsako celico generira zgoščevalni ključ z zgoščevalno funkcijo (angl. hash function). Učinkovitost algoritma je odvisna od unikatnosti generiranega ključa in hitrosti algoritma generiranja ključev. Ob generiranju želimo čim manj kolizij zgoščevalnih ključev. Ključi morajo biti unikatni, saj vsak ključ predstavlja celico v prostoru in ni mogoče, da bi dva ključa opisovala enako celico. Za velikost celice izberemo dolžino gladitvenega jedra *h*.

Zgoščevalna funkcija preslika 2D točko v 1D indeks zgoščevalnega ključa (angl. hash index key) in je definirana kot:

$$hash(\mathbf{r}) = \left(\mathbf{r}_x * p_1 \ \oplus \ \mathbf{r}_y * p_2\right) mod \ n_H, \tag{3.48}$$

kjer je n_H velikost zgoščevalne tabele, **r** pa je definiran kot:

$$\mathbf{r} = \left(\lfloor \mathbf{r}_x / l \rfloor, \lfloor \mathbf{r}_y / l \rfloor\right)^T, \tag{3.49}$$

kjer je *l* dolžina celice. Naslednja parametra pa sta veliki praštevili z vrednostmi:

$$\begin{aligned} p_1 &= 73856093, \\ p_2 &= 19349663. \end{aligned} \tag{3.50}$$

Zgoščevalna tabela mora biti dovolj velika, da ne pride do kolizij zgoščevalnih ključev. Uporabili bomo hevristiko Kelagerja [32], ki definira velikost zgoščevalne tabele z:

$$n_H = prime(2n), \tag{3.51}$$

Kjer je *prime* funkcija, ki vrne naslednje praštevilo večje ali enako od vhodnega parametra in *n* je število delcev v simulaciji. Zaradi definicije gladitvenega jedra, nas zanimajo delci znotraj polmera glajenja *h*, zato definiramo $l = h$, ki predstavlja optimalno izbiro velikosti celice. Prostorska zgoščevalna tabela (angl. spatial hasning table) je na začetku napoljena z vsemi delci. Delec vstavimo v zgoščevalno tabelo z:

$$hash\_table[hash(\mathbf{r}_i)] = \text{delec} \tag{3.52}$$

Kjer je *hash_table* dinamičen seznam, ki lahko hrani več delcev. Ko imamo delce shranjene v seznamu, lahko izvajamo poizvedbe (angl. queries). Vsaka poizvedba najprej izračuna diskretno okolico oz. prostorski obseg delca $\mathbf{r}_Q$ z znano pozicijo in znanim premerom gladitvenega jedra. Diskretno omejeno področje okrog delca je definirano z dvema vozliščema, minimumom in maksimumom, ki ju definiramo kot:

$$BB_{min} = \mathbf{r}\left(\mathbf{r}_Q - (h,h)^T\right) \quad , \quad BB_{max} = \mathbf{r}\left(\mathbf{r}_Q + (h,h)^T\right) \tag{3.53}$$

Zatem preverimo območje od BB_{min} do BB_{max} in ustvarimo unikatno diskretno pozicijo. Za vsako pozicijo pos_D pridobimo dinamični seznam *L*:

$$L = hash\_table[hash(pos_D)] \tag{3.54}$$

Kjer L določimo nič ali več unikatnih delcev. Na koncu preverimo vsak delec *j* v seznamu *L*, če je njegova oddaljenost znotraj dolžine gladitvenega jedra *h*:

$$\left\|\mathbf{r}_Q - \mathbf{r}_j\right\| \leq h. \tag{3.55}$$

4.2 Zaznavanje trkov in obravnavanje mej

Območje simulacije moramo omejiti, drugače se delci premikajo proti neskončnosti. Omejimo jih lahko z eksplicitnimi ovirami (npr. trdne ovire) ali z implicitnimi robovi simulacije (npr. programsko določena meja). Pri simulaciji tekočin robove simulacije po navadi oblikujemo v omejujoče škatle, sfere ali kapsule. Če delec trči z implicitnim robom simulacije, mora ostati znotraj območja. Prav tako ob trku z oviro ne sme prodreti območje ovire.

Obravnavanje trkov lahko razdelimo v dve skupini: zaznavanje trkov (angl. collision detection) in odziv na trk (angl. collision response). Najprej bomo obravnavali zaznavanje trkov med delci okoljem. Predpostavimo, da so vse ovire in meje simulacij fiksne in toge, ki oddajo silo ob trku, same pa jih ne prejmejo. V realnosti zaradi 3. Newtonovega zakona (ali zakon o vzajemnem učinku), ki pravi, da vsako telo deluje na drugo telo z nasprotno enako silo, delci vplivajo tudi na trdne ovire oz. robove simulacij. Če želimo natančno simulacijo interakcije tekočin z okoljem, moramo upoštevati dinamiko togih teles (angl. rigid body dynamics). Območje in ovire v simulacije lahko predstavimo geometrično ali analitično. Pri geometrični predstavitvi ovire običajno moduliramo s trikotniškimi piramidnimi mrežami (angl. tetrahedrons meshes), s katerimi lahko predstavimo poljubni model. Z analitično predstavitvijo opišemo območja oz. like, ki jih lahko zapišemo z matematičnimi formulami z majhno računsko zahtevnostjo. V tej magistrski nalogi bomo implementirali implicitno mejo simulacijske domene v kocki (angl. bounding box) in na kratko predstavili ostale tehnike obravnavanja kolizij.

Delci nosijo pozicijo in hitrost, kar je dovolj informacij za določitev trka. Splošni sistemi za določanje trkov morajo zaznati mesto trka oz. preboja, potem pa s pridobljenimi informacijami ustrezno reagirati z odzivom na trk. Ob trku potrebuje informacije, kot so:

- mesto trka na površju (angl. impact on the surface), $\mathbf{c_p}$,
- globina preboja skozi oviro (angl. penatration depth), d,
- normala površja na točki trka (angl. surface normal), $\mathbf{n}$.

Mesto trka je točka, kjer se je delec dotaknil ovire oz. meje simulacije. Globina preboja je dolžina, ki jo je delec prepotoval ob trku v oviro. Normala površja na točki trka pa je vektor, ki je pravokoten na ravnino meje. Obravnavali bomo samo trke, ki imajo globino kolizije $d > 0$, kar pomeni, da delci na površju objekta ne pridejo v poštev.

Za določitev trka na impliciten rob simulacije je procedura detekcije trka relativno preprosta, pri čemer je določitev točke trka lahko bolj kompleksna operacija (npr. pri sferi ali kapsuli). Pogoste implicitne meje simulacije so krogle, sfere in škatle. Predstavili bomo implementacijo mejnega področja simulacije s pravokotnikom v 2D prostoru. Pravokotnik ni geometrično gladek lik, kot so na primer krog, krogla ali sfera, in ga ni mogoče definirati z zvezno funkcijo.

Uporabimo tehniko projekcije in refleksije (angl. project-and-reflect). Če je delec *i* prebil površje implicitnega roba, njegovo pozicijo spremenimo v smeri projekcije na normalo površja, z magnitudo, ki je proporcialna globini preboja *d*:

$$\mathbf{r}_i = \mathbf{r}_i + d\mathbf{n} \tag{3.56}$$

Kjer je $\mathbf{r}_i = \boldsymbol{cp}$ za implicitne robove simulacije.

Hitrost delca $\mathbf{v}_\mathrm{i}$ se izračuna po standardni vektorski odbojni metodi (angl. standard vector reflection method):

$$\mathbf{v}_i = \mathbf{v}_i - 2(\mathbf{v}_i * \mathbf{n})\mathbf{n} \tag{3.57}$$

Problem zgornje enačbe je popoln elastičen odboj (t.j. kinetične energija se ohrani). V realnosti se tekočine ne odbijajo elastično, zato želimo vnesti nekaj dušenja. Zato enačbo razširimo na:

$$\mathbf{v}_i = \mathbf{v}_i - (1 + c_R)(\mathbf{v}_i * \mathbf{n})\mathbf{n} \tag{3.58}$$

Kjer je koeficient odboja definiran z $0 \leq c_R \leq 1$. Če želimo izničiti hitrost v smeri normale, določimo koeficient $c_R = 0$, ki modelira tekočino brez drsenja (angl. no-slip condition), katere hitrost ob meji je nič. V nasprotnem primeru, ko je $c_R = 1$, dobimo elastičen trk na robovih.

Doseči želimo impulzno kolizijo (angl. impulse-based) točno ob času. Zaradi nastavitev parametrov, lahko enačba (3.58) napačno poveča kinetično energije delca. To preprečimo tako, da omejimo ubežno hitrost delca z:

$$\mathbf{v}_i = \mathbf{v}_i - (1 + c_R \frac{d}{\Delta t \|\mathbf{v}_i\|})(\mathbf{v}_i * \mathbf{n})\mathbf{n}, \tag{3.59}$$

4.3 Parametri simulacije

4.3.1 Parametri tekočin

Kombinacija fizikalnih parametrov določa lastnosti tekočine. Tabela 2 prikazuje parametre, ki so skupne vsem tipom tekočin. Z ustrezno nastavljenimi parametri lahko simuliramo tekočine, kot so voda, plini, sluz itd.

Tabela 2: Lastnosti skupne vsem tekočinam.

Parameter	Simbol	Vrednost	Enota
Gravitacijski pospešek	$\boldsymbol{g}$	$[0, -9{,}82]^T$	m/s^2
Časovni korak	Δt	0.01	s
Temperatura	T	20°	C
Pritisk	P	101325	Pa

Voda je hitra tekočina z nizko viskoznostjo, zato je težko zagotoviti stabilno in realistično simulacijo. Tabela 3 opisuje parametre, ki jih bomo uporabili za simulacijo vode.

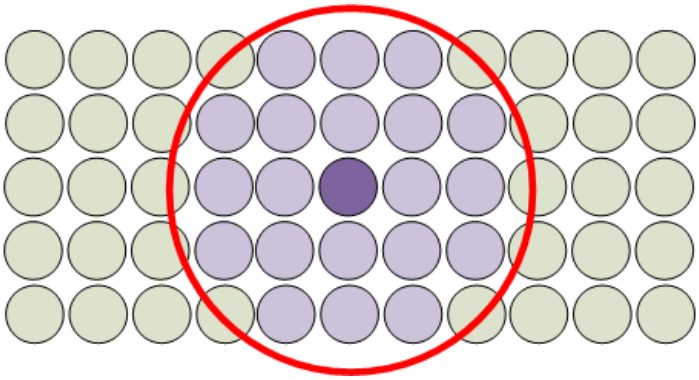
Tabela 3: Parametri za simulacijo vode.

Parameter	Simbol	Vrednost	Enota
Mirujoča gostota	ρ_0	998.29	kg/m^3
Masa delca	M	0,1	Kg
Viskoznost	M	$1.003 * 10^{-3}$	Pa*s
Plinska konstanta	K	3	J
Dolžina glajenja	H	0,2	m

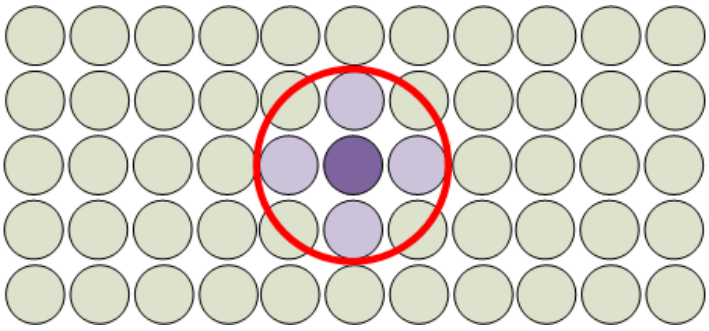
Nekatere parametre je težko določiti v realnosti, zato je potrebno tudi eksperimentiranje z vrednostmi v simulaciji. Ko imamo opravka s fizikalnimi simulacijami, se postopno prilagajanje parametrov (angl. parameter tuning) ne moremo izogniti in ga moramo upoštevati pri načrtovanju rešitev.

4.3.2 Velikost gladitvenega jedra

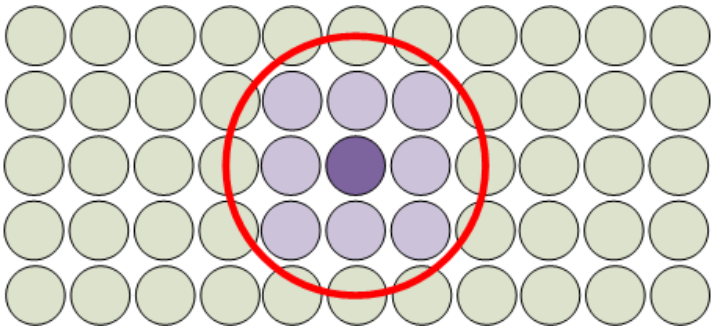
Izbira dolžine glajenja *h* je ključna za stabilno in robustno simulacijo tekočine. Če je parameter *h* prevelik, potem so rezultati simulacije lahko zelo netočni, saj so delci v bližini centra gladitvenega jedra ovrednoteni manj, kot če je parameter *h* manjši. Če pa je parameter h premajhen, potem so lahko rezutati prav tako netočni, saj ni vključenih dovolj delcev v ovrednotenje gladitvenega jedra. Slika 9, Slika 10 in Slika 11 demonstrirajo premajhno, preveliko in optimalno velikost gladitvenega jedra.



Slika 9: Primer prevelikega gladitvenega jedra.



Slika 10: Primer premajhnega gladitvenega jedra.



Slika 11: Primer ustreznega gladitvenega jedra.

Za izbiro optimalne dolžine glajenja *h*, bomo uporabili hevristiko Kelagerja [32], ki poskuša z oceno števila delcev v gladitvenem jedru, izpeljati velikost le tega. Kelager definira naslednji enačbi [32]:

$$x = \left(\frac{n}{V}\right)\frac{4}{3}\pi h^3, \tag{3.60}$$

$$h = \sqrt[3]{\frac{3Vx}{4\pi n}}, \tag{3.61}$$

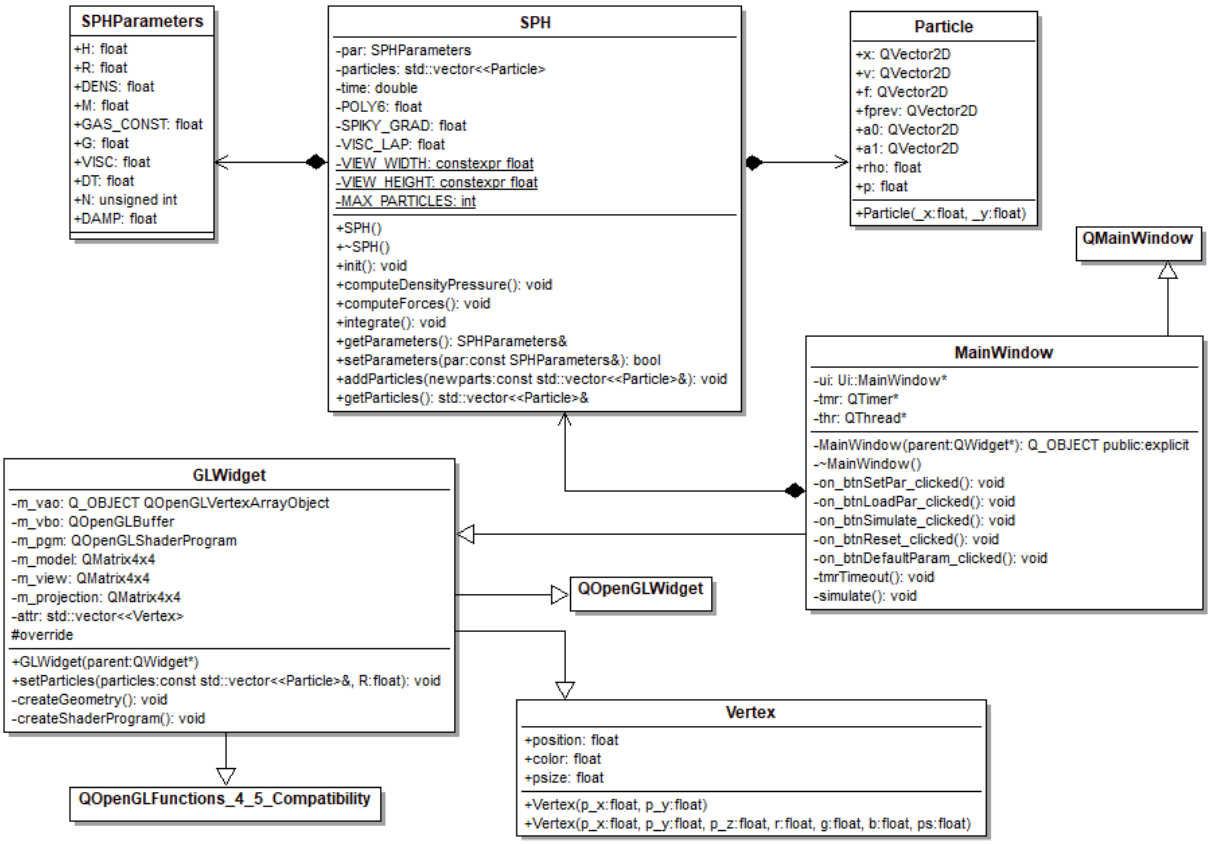
Z enačbo (3.60) določi povprečno število delcev, ki lahko zapolnijo volumen (v 3D) oz. površino (v 2D) gladitvenega jedra. Pri tem *V* predstavlja volumen oz. povšino, *n* pa število delcev, ki ga zavzema. Z obrnjeno enačbo (3.61) pa dobimo dejansko oceno velikosti gladitvenega jedra.

4.3.3 Določitev časovni koraka

Časovni korak Δt je zelo pomemben parameter simulacije, ki določa realni diskretni časovni korak v simulirani tekočini. Če želimo realno-časovno simulacijo, potem časovni korak določa število sličic na sekundo (angl. frame per seconds, FPS) z $1/\Delta t$ Hz. V aplikaciji želimo doseči čim večji časovni korak, s katerim še lahko zadostimo zahtevam simulacije. Izberemo fiksen časovni korak 10 ms, ki zagotavlja 100 FPS. Časovno integracijo bomo izvedli z algoritmom žabjega preskoka, ki smo ga podrobneje opisali v poglavju 3.4.3. Algoritem žabjega preskoka se boljše obnese v primerjavi z Eulerjevo metodo in Verletovo metodo v stabilnosti in učinkovitosti. Fizikalni parametri, ki so odvisni od časovnega koraka (npr. plinska konstanta), morajo biti ustrezno adaptirani, da ne privedejo do nestabilnosti simulacije.

4.3.4 Implementacija algoritma

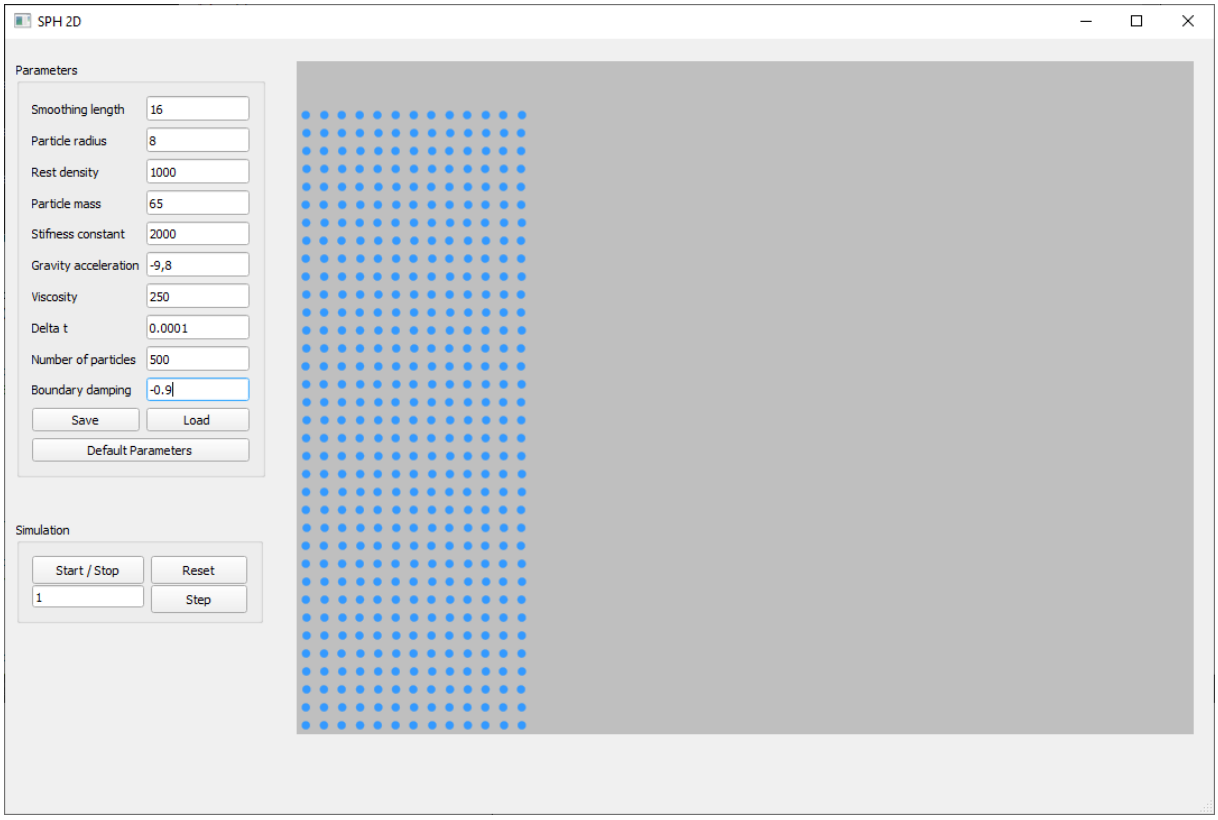
Program je napisan v programskem jeziku C++ in programskim okoljem Qt. Gre za večplatformno programsko okolje, ki v osnovi omogoča razvoj grafičnih uporabniških vmesnikov, ob tem pa še podpira kopico knjižnic za delo s komunikacijskimi tehnologijami, večpredstavnostmi, bazami podatkov, spletnimi tehnologijami itd. Na sliki Slika 12 vidimo predstavitev diagrama UML implementiranega programa. V začetni funkciji *main* inicializiramo programsko okolje Qt, nastavimo vrednosti programskega okolja OpenGL, inicializiramo razred programa *MainWindow*, ki ga lahko vidimo na diagramu UML, in začnemo z izvajanjem glavne zanke procesiranja dogodkov (angl. event loop) okolja Qt.



Slika 12: Diagram UML izvedenega programa

Slika 13 prikazuje zasnova grafičnega vmesnika program v začetnem stanju. Uporabnik nastavi želene parametre in požene simulacijo s klikom na gumb »Start / Stop«. S ponovnim pritiskom na ta gumb simulacijo tudi ustavi. Ob pritisku na gumb za začne izvajat metoda *simulate*, ki je del razreda *MainWindow*. Znotraj te metode se pokličejo naslednje funkcije razreda SPH:

1. *computeDensityPressure*, ki izračuna gostoto delcev (implementacijo vidimo na sliki Slika 15),
2. *computeForces*, ki izračuna sile pritiska, viskoznosti in gravitacijo, ki vplivajo na delce (implementacijo vidimo na sliki Slika 16),
3. *integrate*, ki z metodo žabjega preskoka izvede numerično integracijo simulacije (implementacijo vidimo na sliki Slika 17).



Slika 13: Zasnova grafičnega vmensika z izvedeno metodo SPH v začetnem stanju

Slika 14 prikazuje psevdokod program in naslednji seznam točk opiše izvajanje algoritma po korakih:

1. začetna inicializacija sistema,
2. ponavljamo naslednje korake do konca simulacije za vse delce:
   2.1. poiščemo sosede obravnavanega delca,
   2.2. izračunamo gostote delca z enačbo (3.2),
   2.3. izračunamo pritisk delca iz njegove gostote z enačbo (3.33),
   2.4. izračunamo silo pritiska delca z enačbo (3.31),
   2.5. izračunamo silo viskoznosti delca z enačbo (3.35),
   2.6. izračunamo zunanje (gravitacijske) sile na delec z enačbo za težni pospešek in
   2.7. numerično integriramo z metodo žabjega preskoka,
3. vizualizacija delcev z OpenGL.

inicializacija programa

for all delec i **do**

poišči sosede j z NNS

for all delec i **do**

$\rho_i = \sum_j m_j W_{ij}$

izračunaj p_i iz ρ_i

for all delec i **do**

$\mathbf{F}_i^{pritisk} = -\frac{m_i}{\rho_i} \nabla p_i$

$\mathbf{F}_i^{viskoznost} = m_i \nu \nabla^2 \mathbf{v}_i$

$\mathbf{F}_i^{gravitacija} = m_i \mathbf{g}$

$\mathbf{F}_i(t) = \mathbf{F}_i^{pritisk} + \mathbf{F}_i^{viskoznost} + \mathbf{F}_i^{gravitacija}$

for all delec i **do**

$\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t) + \Delta t \mathbf{F}_i(t)/m_i$

$\mathbf{x}_i(t + \Delta t) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t + \Delta t)$

zaznavanje trkov in obravnavanje mej

vizualizacija delcev

Slika 14: Psevdokod zasnovanega algoritma za simulacijo vode z metodo SPH.

V koraku inicializacije pripravimo program na izvajanje simulacije z metodo SPH. Naslednji koraki prikazujejo postopek simulacije:

1. uvodno nastavi programsko okolje Qt in ustvari okno OpenGL za prikazovanje 2D grafike,
2. ustvari sistem delcev z začetnimi pozicijami in hitrostmi. Število delcev je nastavljivo,
3. naloži privzete parametre sistema delcev,
4. uvodno nastavi numerično integracijo žabjega preskoka za vse delce.

Metoda SPH operira z delci, ki jih lahko enostavno vizualiziramo v 2D (krogi) in 3D (sfere) prostoru. Možnost sledenja posameznim delcem olajša prepoznavo toka tekočine. Uporabniki končnih aplikacij, ki pričakujejo naravni tok tekočine, pa se ne zadovoljijo s to rešitvijo. Za vizualizacijo tekočin moramo uporabiti tehnike, ki generiranjo geometrijo prostega površja (angl. free surface) iz gruč delcev. V to skupino spadaja metode:

- meta krogle (angl. metaballs) [33],
- korakajoče kocke (angl. marching cubes) [34],
- sploščenje površij (angl. surface splatting) [35] in
- tehnika preproge (angl. carpet visualization).

V tej magistrski nalogi bomo za vizualizacijo zraven osnovnega izrisa delcev uporabili algoritem sploščenja površij.

```
void SPH::computeDensityPressure()
{
    for(auto &pi : particles) {
        pi.rho = 0.0;
        for(auto &pj : particles) {

            // NOTE: calculation of vector 'rij' norm can be omited
            QVector2D rij = pj.x - pi.x;
            float rsq = rij.lengthSquared();
            float hsq = pow(par.H, 2);

            if(rsq < hsq) {
                pi.rho += par.M * POLY6 * pow(hsq-rsq, 3.0f);
            }
        }
        pi.p = par.GAS_CONST*(pi.rho - par.DENS);
    }
}
```

Slika 15: Implementacija računanja gostote

```
void SPH::computeForces()
{
    for(auto &pi : particles) {

        pi.fp = QVector2D(0.0, 0.0); // Pressure force vector
        pi.fv = QVector2D(0.0, 0.0); // Viscosity force vector
        pi.fg = QVector2D(0.0, par.G * pi.rho); // Force of gravity

        for(auto &pj : particles) {
            if(&pi == &pj)
                continue;

            QVector2D rij = pj.x - pi.x;
            float r = rij.length();

            if(r < par.H) {
                // Compute pressure force contribution
                pi.fp += rij.normalized()*par.M*(pi.p + pj.p)/(2.0f *
pj.rho) * SPIKY_GRAD*pow(par.H-r, 2.0f);
                // Compute viscosity force contribution
                pi.fv += par.VISC*par.M*(pj.v - pi.v)/pj.rho *
VISC_LAP*(par.H-r);
            }
        }
```

```
        pi.fp = pi.fp*-1;
        pi.f  = pi.fp + pi.fv + pi.fg;
        pi.a  = pi.f / pi.rho;
    }
}
```

Slika 16: Implementacija računanja sil

```
void SPH::integrate()
{
    for(auto &p : particles) {
        // Forward Euler integration
        //p.x += par.DT * p.v;
        //p.v += par.DT * p.a1;

        // Leapfrog integration
        p.v += 0.5 * p.a * par.DT;
        p.x += p.v * par.DT;

        // Velocity Verlet
        //p.x = p.x + p.v*par.DT + p.a0*pow(par.DT, 2)/2;
        //p.v = p.v + (p.a0+p.a1)/2 * par.DT;

        // Check boundary of the simulation.
        if(p.x[0] - BOR_DIST < 0.0f) {
            p.v[0] *= par.DAMP;
            p.x[0]  = BOR_DIST;
        }
        if(p.x[0] + BOR_DIST > VIEW_WIDTH) {
            p.v[0] *= par.DAMP;
            p.x[0]  = VIEW_WIDTH - BOR_DIST;
        }
        if(p.x[1] - BOR_DIST < 0.0f) {
            p.v[1] *= par.DAMP;
            p.x[1]  = BOR_DIST;
        }
        if(p.x[1] + BOR_DIST > VIEW_HEIGHT) {
            p.v[1] *= par.DAMP;
            p.x[1]  = VIEW_HEIGHT - BOR_DIST;
        }
    }

    time += par.DT;
    qDebug("SPH time: %f", time);
}
```

Slika 17: Implementacija integracije in obravnavanja mej

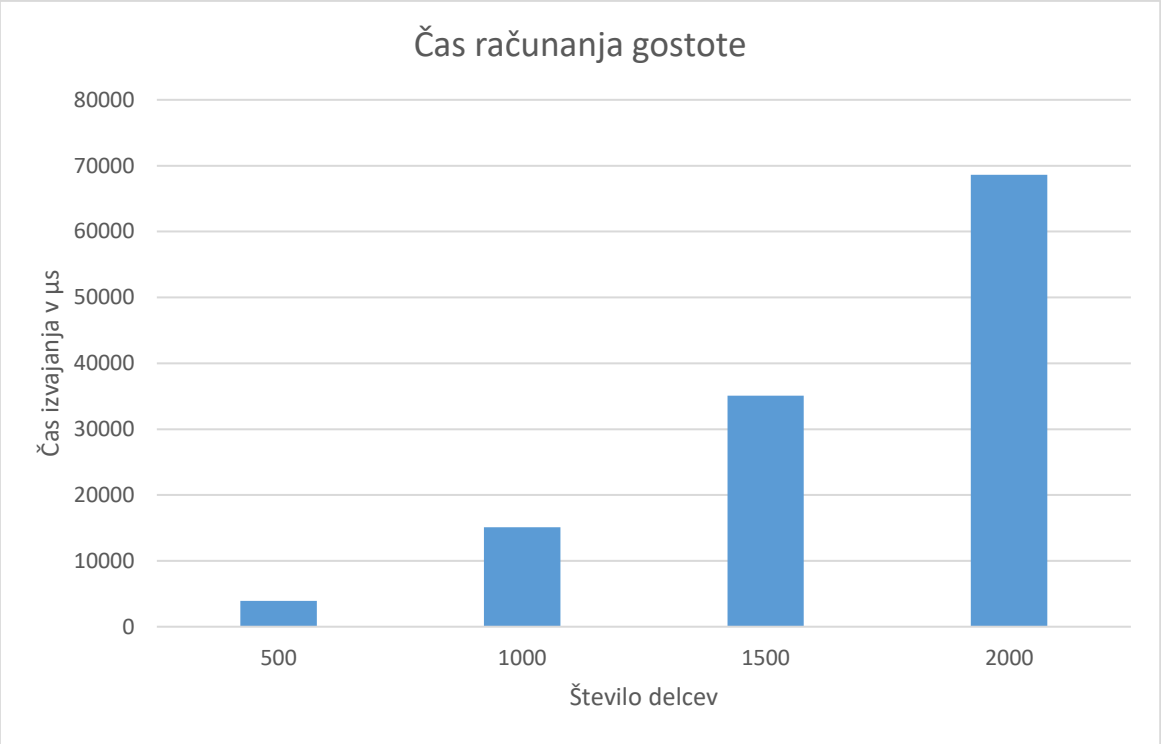
5 REZULTATI

Algoritem hidrodinamike zglajenih delcev lahko analiziramo na več načinov. Osredotočimo se lahko na fizikalno natančnost simulacije z validacijskimi testi [36] ali na učinkovitost izvajanja algoritma pri obremenitvi z različnim številom delcev [32]. Ker smo se v tej magistrski nalogi omejili pri izvedbi metode SPH z 2D prostorom, neupoštevanjem turbolenc in površinskih napetosti, rezultati niso popolnoma fizikalno točni, ampak gre samo za približek simulacije tekočine. Zato se bomo v tem poglavju osredotočili na učinkovitost metode SPH pri različnih obremenitvah.

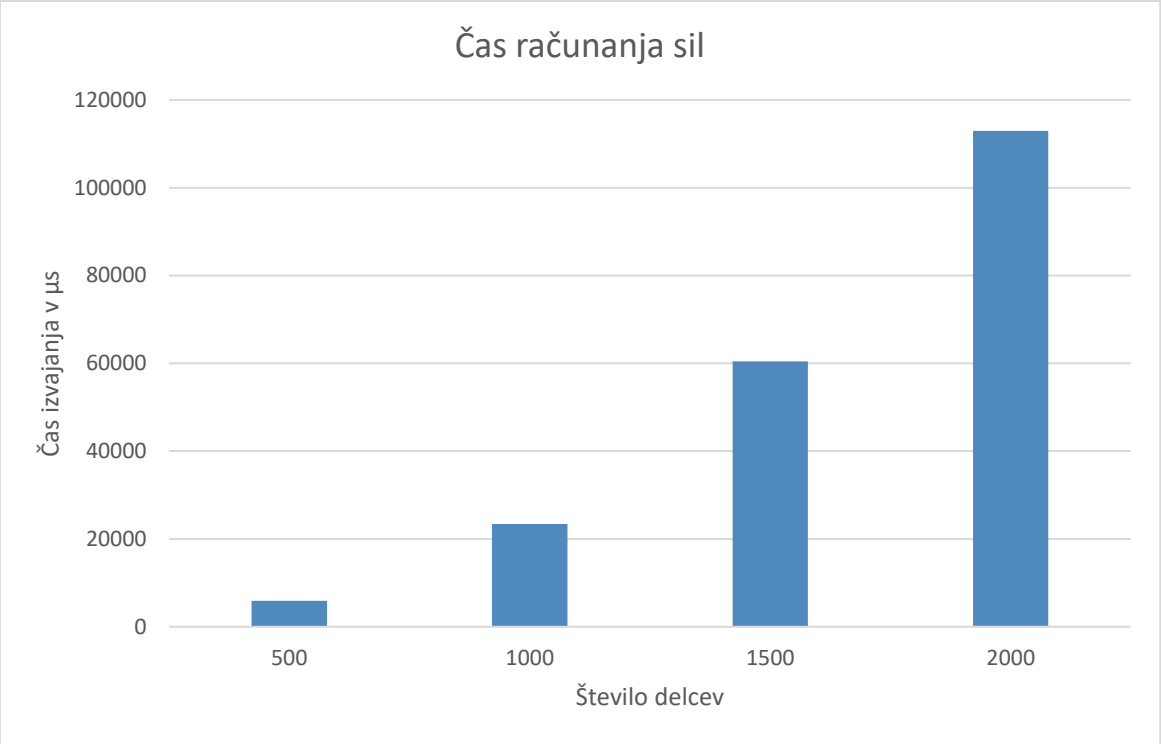
Z implementiranim programom smo opravili analizo zmogljivosti, v kateri smo preverili zmogljivost izvajanja kompleksnih simulacij. Uporabili smo parametre simulacije, ki smo jih definirali v prejšnjem poglavju. Program smo izvajali na štirijedrnem procesorji Intel Core i7-6700. V tabeli Tabela 4 vidimo povprečno hitrost izvajanja posameznih delov algoritma. Opazimo, da se čas izvajanja proporcionalno povečuje s številom delcev simulacije.

Tabela 4: Izmerjeni časi računanja posameznih delov izvedbe algoritma

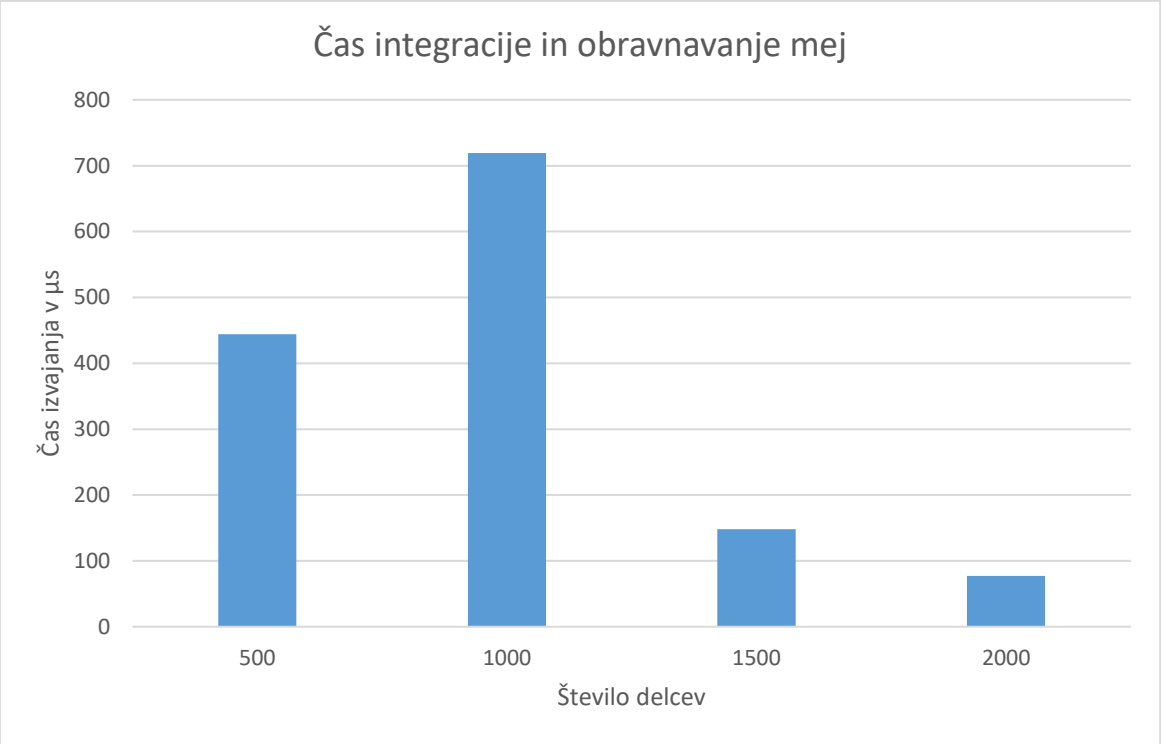
Število delcev (n)	**Čas računanja gostote (μs)**	**Čas računanja sil (μs)**	**Čas integracije (μs)**
500	3904	5941	444
1000	15119	23431	719
1500	35097	60465	148
2000	68631	112935	77



Slika 18: Čas računanja gostote v izvedbi metode SPH



Slika 19: Čas računanja sil v izvedbi metode SPH



Slika 20: Čas integracije in obravnavanja mej v izvedbi metode SPH

6 SKLEP

V okviru magistrske naloge smo preučili algoritem hidrodinamike zglajenih delcev v 2D prostoru. Metoda SPH, kot se algoritem hidrodinamike zglajenih delcev imenujemo krajše, je postopek, s katerim simuliramo dinamiko tekočin. Metoda SPH je bila primarno razvita za probleme astrofizike, kasneje pa je bila adaptirana tudi za probleme iz ostalih področij. Predvsem se je uveljavila za simulacijo tekočin.

V magistrskem delu smo preučili metodo SPH iz teoretskega vidika in implementirali praktično izvedbo simulacije tekočine v 2D prostoru. Rezultate smo analizirali iz vidika računske zahtevnosti in ugotovili, da se čas izvajanja algoritma povečuje proporcionalno s številom simuliranih delcev.

LITERATURA

[1] R. Bridson and C. Batty, “Computational Physics in Film,” *Science (80-.).*, vol. 330, no. 6012, p. 1756 LP-1757, Dec. 2010.

[2] M. Müller, D. Charypar, and M. Gross, “Particle-Based Fluid Simulation for Interactive Applications,” *Proc. 2003 ACM SIGGRAPH/Eurographics Symp. Comput. Animat.*, no. 5, pp. 154–159, 2003.

[3] M. Desbrun and M.-P. Gascuel, “Smoothed Particles: A new paradigm for animating highly deformable bodies,” Springer, Vienna, 1996, pp. 61–76.

[4] D. L. Tonnesen and D. Love, *Dynamically coupled particle systems for geometric modeling, reconstruction, and animation*. University of Toronto, Dept. of Computer Science, 1998.

[5] J.-C. Lombardo and C. Puech, “Oriented Particles: A Tool for Shape Memory Objects Modelling,” pp. 255–262, May 1995.

[6] A. P. Witkin and P. S. Heckbert, “Using particles to sample and control implicit surfaces,” in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques - SIGGRAPH ’94*, 1994, pp. 269–277.

[7] D. Stora, P.-O. Agliati, M.-P. Cani, F. Neyret, and J.-D. Gascuel, “Animating Lava Flows,” pp. 203--210, 1998.

[8] J. Stam, “Stable fluids,” in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques - SIGGRAPH ’99*, 1999, pp. 121–128.

[9] N. Foster and R. Fedkiw, “Practical animation of liquids,” in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH ’01*, 2001, pp. 23–30.

[10] D. Enright, S. Marschner, R. Fedkiw, D. Enright, S. Marschner, and R. Fedkiw, “Animation and rendering of complex water surfaces,” in *Proceedings of the 29th annual conference on Computer graphics and interactive techniques - SIGGRAPH ’02*, 2002, vol. 21, no. 3, p. 736.

[11] T. Takahashi, H. Ueki, A. Kunimatsu, and H. Fujii, “The simulation of fluid-rigid body interaction,” in *ACM SIGGRAPH 2002 conference abstracts and applications on - SIGGRAPH ’02*, 2002, p. 266.

[12] D. Nixon and R. Lobb, “A fluid-based soft-object model,” *IEEE Comput. Graph. Appl.*, vol. 22, no. 4, pp. 68–75, Jul. 2002.

[13] M. Carlson, P. J. Mucha, R. B. Van Horn, and G. Turk, “Melting and flowing,” in

Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation - SCA '02, 2002, p. 167.

[14] W. T. Reeves, "Particle systems---a technique for modeling a class of fuzzy objects," in *Proceedings of the 10th annual conference on Computer graphics and interactive techniques - SIGGRAPH '83*, 1983, vol. 17, no. 3, pp. 359–375.

[15] M. Ihmsen, J. Cornelis, B. Solenthaler, C. Horvath, and M. Teschner, "Implicit Incompressible SPH," pp. 1–11.

[16] R. A. Gingold and J. J. Monaghan, "Smoothed particle hydrodynamics: theory and application to non-spherical stars," *Mon. Not. R. Astron. Soc.*, vol. 181, no. 3, pp. 375–389, Dec. 1977.

[17] L. B. Lucy and L. B., "A numerical approach to the testing of the fission hypothesis," *Astron. J.*, vol. 82, p. 1013, Dec. 1977.

[18] D. Violeau, "Fluid mechanics and the SPH method: Theory and applications," *Spat. Epidemiol. Methods Appl.*, no. July, pp. 87–103, 2000.

[19] J. W. Swegle, D. L. Hicks, and S. W. Attaway, "Smoothed particle hydrodynamics stability analysis," *J. Comput. Phys.*, 1995.

[20] M. Becker, H. Tessendorf, and M. Teschner, "Direct Forcing for Lagrangian Rigid-Fluid Coupling," *IEEE Trans. Vis. Comput. Graph.*, vol. 15, no. 3, pp. 493–503, May 2009.

[21] M. Müller, B. Solenthaler, R. Keiser, and M. Gross, "Particle-Based Fluid-Fluid Interaction," 2005.

[22] J. P. Morris, "Simulating surface tension with smoothed particle hydrodynamics," *Int. J. Numer. Methods Fluids*, 2000.

[23] L. Verlet, "Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules," *Phys. Rev.*, vol. 159, no. 1, pp. 98–103, Jul. 1967.

[24] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3rd ed. New York, NY, USA: Cambridge University Press, 2007.

[25] W. C. Swope, H. C. Andersen, P. H. Berens, and K. R. Wilson, "A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters," *J. Chem. Phys.*, vol. 76, no. 1, pp. 637–649, Jan. 1982.

[26] "SOLIDWORKS Flow Simulation." [Online]. Available: https://www.solidworks.com/product/solidworks-flow-simulation. [Accessed: 02-Dec-

2018].

[27] “GameWorks PhysX Overview | NVIDIA Developer.” [Online]. Available: https://developer.nvidia.com/gameworks-physx-overview. [Accessed: 02-Dec-2018].

[28] “RealFlow.” [Online]. Available: http://www.nextlimit.com/realflow/. [Accessed: 02-Dec-2018].

[29] M. De Berg, O. Cheong, M. Van Kreveld, and M. Overmars, *Computational geometry: Algorithms and applications*. 2008.

[30] N. Lukač and B. Žalik, “Fast Approximate k-Nearest Neighbours Search Using GPGPU,” in *GPU Computing and Applications*, Singapore: Springer Singapore, 2015, pp. 221–234.

[31] M. Teschner, B. Hiedelberger, M. Müller, D. Pomeranets, and M. Gross, “Optimized Spatial Hashing for Collision Detection of Deformable Objects,” *Vis. Model. Vis.*, 2003.

[32] M. Kelager, “Lagrangian Fluid Dynamics Using Smoothed Particle Hydrodynamics,” *Camb. Monogr. Mech.*, vol. 77, 2006.

[33] J. F. Blinn, “A Generalization of Algebraic Surface Drawing,” *ACM Trans. Graph.*, vol. 1, no. 3, pp. 235–256, 1982.

[34] W. E. Lorensen and H. E. Cline, “Marching cubes: A high resolution 3D surface construction algorithm,” *ACM SIGGRAPH Comput. Graph.*, 1987.

[35] M. Zwicker, H. Pfister, J. van Baar, and M. Gross, “Surface splatting,” in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH ’01*, 2001.

[36] “SPHeric - Validation Tests.” [Online]. Available: http://spheric-sph.org/validation-tests. [Accessed: 09-Dec-2018].