SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 475

Animacija nestlačivih fluida temeljena na metodi SPH

Bruno Mikuš

Zagreb, lipanj 2012.

Sadržaj

1 Uvod

Povijest simulacije fluida

Matematičko modeliranje i fizikalno opisivanje fluida već je dugo vremena interesantan problem matematičarima i fizičarima diljem svijeta. Razvojem industrije, pomorstva, astronomije, geografije i mnogih drugih disciplina postalo je interesantno matematičke modele fluida primijeniti na stvarne sustave i riješiti probleme poput aerodinamike, hidrodinamike, nastanka i kretanja zvijezda, projektiranja vozila te mnogih drugih. Rezultate bilo kakve simulacije najlakše je interpretirati ako se prikažu što je zadatak računalne grafike i u području simulacije fluida, riječ je naravno o animaciji fluida.

Uz inženjerska područja, potreba za animiranim fluidima nastala je i u zabavnoj industriji. Tako animirani filmovi novog doba uz funkcionalnost fluida zahtijevaju i dobru vizualizaciju. U prvim animiranim filmovima koji su koristili simulacije fluida, poput filma „Shrek“, računalno najzahtjevnije scene bile su upravo one s tekućinama. S porastom dostupne računalne moći u kućanstvima, nastala je potreba industrije igara za sustavima simulacija fluida pa npr. „Alice: MadnessReturns“ koristi Nvidia-inPhysX simulator za fluide. Glavni zahtjev takvih simulacija je izvodivost u stvarnom vremenu i u današnje doba najviše istraživanja i razvoja simulacije fluida ide u tom smjeru.

Metode simulacije fluida

Općeniti postupak simulacije fluida može se opisati na sljedeći način. Fluid (plin ili tekućina) je sastavljen od mnoštva čestica koje međusobno djeluju raznima silama (tlak, viskoznost, napetost, turbulencije). Prostor u kojem se fluid nalazi je moguće zamisliti kao veliku pravilnu rešetku čije svako presjecište predstavlja točku uzorkovanja fluida (to je poznatije kao Eulerova rešetka) ili kao prazan prostor kojeg ispunjavaju nakupine molekula fluida – čestice koje vrše međusobnu interakciju (što je poznatije kao Lagrangian metoda). Sile unutar fluida određuju se na temelju stanja prostora koji fluid zauzima i na temelju svojstava tog fluida (tlak, temperatura, gustoća, brzina). Prema jednadžbama koje opisuju ponašanje fluida, kao što su Navier-Stokesove jednadžbe, vrši se integracija sila u fluidu. Iz sila se određuje akceleracija, brzina i pomak za neki vremenski korak i postupak se ponavlja.

Iako djeluje jednostavno, problem je u velikoj vremenskoj složenosti za sve postupke koji pravilno vrše simulaciju. Neke metode nikada neće biti pogodne za

korištenje u stvarnom vremenu, dok su neke metode idealne i za interaktivne sustave i za matematičke simulacije. Riječ je prvenstveno o metodi hidrodinamike izglađenih čestica (SPH, *engl. Smoothed Particle Hydrodynamics*) koja je i tema ovog rada. Ideja metode je koristiti sustav čestica koje djeluju jedna na drugu ovisno o međusobnoj udaljenosti. Svojstva i utjecaji čestica se izglađuju korištenjem posebnih funkcija – jezgri. Koliko je ova metoda efikasna govori činjenica da su sustavi simulacije fluida u popularnom alatu za 3D modeliranje (Blender) i alatu za simuliranje fizike (Nvidia PhysX) temeljeni upravo na SPH metodi.

Nakon simulacije, fluid je potrebno vizualizirati za što postoji nekoliko tehnika, a ovisi o potrebama sustava koja će biti najpogodnija. Vizualizacija se svodi na problem računanja i prikaza izopovršine fluida pri čemu se koriste kombinacije više algoritama. Generirana površina može biti eksplicitna u obliku poligonalne mreže i tada je njen prikaz jednostavan ili može biti implicitna što znači da je potreban dodatan korak pretvorbe u poligonalnu mrežu ili korištenje neke metode prikaza implicitne površine u prostoru pogleda (*engl. screen space rendering*).

1.1 Pregled područja

Hidrodinamika izglađenih čestica

Korištenje prostorne rešetke predstavljalo je problem astrofizičarima prije četiri desetljeća jer je računanje konačnih razlika i integrala na velikom broju točaka rešetke bilo nepraktično, kompleksno i ograničeno. S tim ciljem preuzet je Lagrangianov čestični opis fluida koji je fokusiran na elemente fluida, a ne na prostornu rešetku u kojoj se fluid nalazi. Korištenjem statističke metode izglađujućih jezgri i tehnike delta krivulje računa se distribucija gustoće fluida, a pozicija čestica se mijenja na temelju izračunate gustoće. Ta metoda nazvana je hidrodinamika izglađenih čestica (Gingold, 1977). Do sličnih je jednadžbi došao i Lucy (1977.) prilikom opisa testiranja hipoteze o fisiji kao metodi nastanka binarnih zvjezdanih sustava. Opsežan i sustavan pregled SPH metode napravio je Monaghan (1992.) te 15 godina nakon nastanka metode navodi mnoga područja astrofizike u kojima je SPH našao primjenu poput binarnih sustava, sudara zvijezda, nastanka mjeseca i problema sudara, eksplozija supernove, kretanje blizu crne rupe i slično. Osim u astrofizici, SPH metodu su primjenjivali i za opisivanje dinamike plinova i računanje nestlačivih tokova.

Među prvim pojavljivanjima u računalnoj grafici SPH metoda je korištena za potrebe simulacije vatre i dima (Stam, 1995.) te za simuliranje tijela koja podnose iznimne deformacije (Desbrun, 1996.). Za razliku od prethodnih tehnika, temeljenih na Eulerovoj prostornoj rešetci, SPH metoda je omogućila realističan prikaz fluida uz interaktivne brzine računanja i crtanja. Osim rješavanja sila pritiska i viskoznosti iz Navier-Stokesovih jednadžbi, modelirane su i sile napetosti površine, dizajnirane posebne izglađujuće jezgre i predložene metode vizualizacije (Müller, 2003.). U kasnijim je radovima ostvareno i međudjelovanje više tekućina, stvaranje mjehurića zraka zbog sila napetosti površine unutar fluida i promjene agregatnih stanja pri promjeni temperature (Müller, 2005.). U zadnjih par godina naglasak razvoja SPH metode je na paralelizaciji i dodatnom ubrzanju postupka (Goswami, 2010.).

Lagrangian metode

Postoji još nekoliko Lagrangian metoda koje je bitno spomenuti. Polu-implicitna metoda pokretnih čestica slična je SPH metodi, a razlikuje se pri računanju gradijenta i traženju rješenja diferencijalnih jednadžbi toka fluida te ima prednost nad SPH metodom pri rješavanju nestlačivog toka (Koshizuka, 1996.). Druga interesantna metoda koristi sličan pristup kao SPH metoda prilikom računanja gustoće, a razlika je što se prilikom računanja pomaka čestica fluida vrši ujednačavanje tlaka između čestica te se pri računanju viskoznosti koriste opruge za simulaciju elastičnosti i plastičnosti te linearnih i kvadratnih impulsa temeljenih na razlici brzina (Clavet, 2005.).

Metode Eulerove rešetke

Osim simulacije fluida Lagrangian metodama, fluid je moguće simulirati koristeći Eulerovu rešetku. Problem ovog pristupa je velika prostorna složenost, nedostatak detalja u rešetci, teško miješanje različitih fluida i problem nestabilnosti za velike vremenske korake. Problem nestabilnosti za veće vremenske korake riješen je kombinacijom s čestičnim pristupom u semi-Lagrangian metodi, ali je uvedena disipacija mase koja nije pogodna za animaciju fluida (Stam, 1999.). Problem praćenja površine u Eulerovoj rešetci najefikasnije je riješen korištenjem implicitne funkcije razinskog skupa (*engl. level set function*) čija je vrijednost manja od 0 na mjestima gdje se nalazi fluid (Foster, 2001.). Metoda razinskog skupa je razvijena u metodu čestičnog razinskog skupa, a dodatno poboljšava praćenje površine, stabilnost simulacije i sprječava nestajanje mase fluida

(Enright, 2002.). Čak i uz korištenje programa za sjenčanje i protočne arhitekture modernih grafičkih procesora za simuliranje fluida, metode koje koriste rešetku ne odlikuju se velikom brzinom računanja. Na primjer, za rešetku reda veličine 128^3 ćelija, potrebno je $\frac{1}{8}$ sekunde za izračun jednog vremenskog koraka fluida tada dostupnim računalima (Matsuda, 2007.) što iako daje impresivne rezultate, nije primjenjivo za sustave u stvarnom vremenu. Rješenje za simulaciju u stvarnom vremenu je korištenje mogućnosti paralelnog računanja na GPU (Crane, 2007.).

Vizualizacija fluida

Pristupi prikazu fluida su različiti te postoji više uspješno korištenih metoda. Poligonalnu mrežu površine fluida najjednostavnije je dobiti algoritmom pokretnih kocki i takvu je mrežu jednostavno prikazati, a sam algoritam pogodan je i za paralelizaciju na GPU (Dyken, 2008.). Za dodatno ubrzanje razvijene su metode prikaza poligonalnom mrežom u prostoru pogleda koja se generira iz visinske mape pomoću algoritma pokretnih kvadra (Müller, 2007.). Implicitnu površinu fluida koja se dobije algoritmom poput metakugli moguće je vizualizirati bacanjem zrake, a za manji broj metakugli postižu se i interaktivne brzine iscrtavanja (Kanamori, 2008.). Skup točaka površine fluida može predstavljati skup surfela koji se mogu rasterizirati kao male elipse uz dobre brzine iscrtavanja bez posebne paralelizacije i iskorištavanja snage GPU-a(Pfister, 2000.). Bolju vizualizaciju SPH fluida moguće je ostvariti prilagodbom oblika čestice koja je umjesto sferom prikazana elipsoidom tj. anizotropnom jezgrom (Yu, 2010.).

2 Dinamika fluida

Dinamika fluida je podskup mehanike fluida koji se bavi tokom - gibanjem fluida. Računalna dinamika fluida (*CFD, engl. Computational fluid dynamics*) je nadogradnja nad dinamiku fluida koja primjenjuje numeričke metode za rješavanje tokova fluida.

2.1 Fizika dinamike fluida

Ponašanje toka fluida moguće je opisati na nekoliko načina:

- stlačivi i nestlačivi tok – ovise o promjeni gustoća fluida, za nestlačivi tok vrijedi da je promjena gustoće u vremenu jednaka nuli
- viskozni i neviskozni tok – ovise o utjecaju trenja unutar fluida
- za stacionaran tok brzina i tlak su samo funkcije položaja, a za nestacionaran tok brzina i tlak su funkcije položaja i vremena
- laminaran i turbulentan tok – ovise o postojanju vrtložnih gibanja
- subsoničan, transsoničan, supersoničan i hipersoničantok – ovise o Machovom broju. Ova tipizacije je posebno bitna za aerodinamiku.

Za potrebe ovog rada fluidi koji će biti razmatrani opisuju nestlačiv, viskozan, stacionaran, laminaran i subsoničan tok (Machov broj manji od 0.3).

Parametri koji definiraju fluid su gustoća mirovanja, koeficijent viskoznosti, plinska konstanta i molarna masa, a svojstva fluida koja se mijenjaju i računaju kao funkcija prostora i vremena su gustoća, tlak, temperatura i brzina.

Temeljni zakoni koji upravljaju ponašanjem fluida su:

- zakon o očuvanju mase – ukupna masa izoliranog sustava je konstantna
- zakon očuvanja količine gibanja –u zatvorenom sustavu promjena momenta između dva objekta je ista ili suprotna, ovaj zakon bitan je pri sudarima
- zakon očuvanja energije – ukupna količina energije unutar izoliranog sustava je konstantna

Fluid se u konačnici promatra kao skup sudarajućih čestica na koje se primjenjuju navedeni zakoni, a njegovo se ponašanje opisuje diferencijalnim jednadžbama od kojih su Navier-Stokesove jednadžbe korištene u gotovo svim CFD sustavima. Pojednostavljenje tih jednadžbi daje Eulerove jednadžbe koje se koriste za neviskozan tok.

2.2 Navier-Stokesove jednadžbe

Navier-Stokesove jednadžbe su u svojoj osnovi jednadžbe očuvanja količine gibanja. Stoga je osnovna jednadžba za nestlačive tokove jednadžba gibanja:

$$\rho\left(\frac{\partial\vec{u}}{\partial t}+\vec{u}\cdot\nabla\vec{u}\right)=-\nabla p+\mu\nabla^2\vec{u}+\vec{f} \tag{2-1}$$

Ovdje je ρ gustoća fluida, $\vec{u}$ je brzina toka fluida, p je tlak, μ je dinamička viskoznost fluida, a $\vec{f}$ su sile koje djeluju na fluid (gravitacija, centrifugalna sila, centripetalna sila i slično). Lijevi dio izraza odnosi se na inerciju volumena fluida i predstavlja promjenu brzine toka (akceleraciju), dok desni dio izraza predstavlja sile koje utječu na promjenu fluida:

- $-\nabla p$ - sila tlaka koja usmjerava fluid suprotno od područja rasta tlaka
- $\mu\nabla^2\vec{u}$ - sila viskoznosti stvara težnju ujednačavanja brzina unutar fluida
- $\vec{f}$ - vanjske sile

Kako bi bio zadovoljen zakon o očuvanju mase za fluid, potrebno je sustavu dodati i jednadžbu kontinuiteta mase:

$$\frac{\partial\rho}{\partial t}+\nabla\cdot(\rho\vec{u})=0 \tag{2-2}$$

Nestlačivi tok je svaki tok u kojem se gustoća fluida ne mijenja, što se izražava kao parcijalna derivacija gustoće po vremenu jednaka nuli:

$$\frac{\partial\rho}{\partial t}=0 \tag{2-3}$$

Jednadžba kontinuiteta (2-2) tada prelazi u:

$$\nabla\cdot(\rho\vec{u})=0$$

Odnosno u:

$$\nabla\vec{u}=0 \tag{2-4}$$

Matematički izraz (2-4) govori kako oko volumena fluida ne postoji promjena brzine koja izlazi izvan volumena fluida odnosno kako ne postoje čestice (mase) koje napuštaju fluid.

Uvrštavanjem (2-4) u (2-1) dolazi se do jednostavnije jednadžbe momenta:

$$\rho\frac{\partial\vec{u}}{\partial t}=-\nabla p+\mu\nabla^2\vec{u}+\vec{f} \tag{2-5}$$

Jednadžbama (2-1) i (2-4) nedostaje jedino jednadžba očuvanja energije, a ona glasi:

$$\rho\left(\frac{\partial \varepsilon}{\partial t} + \vec{u} \cdot \nabla \varepsilon\right) = \nabla \cdot (K_h \nabla T) - p \nabla \cdot \vec{u} \tag{2-6}$$

Ovdje je ε termodinamička unutarnja energija, T je temperatura, a K_h je koeficijent toplinske vodljivosti. U računalnim simulacijama ova se jednadžba javlja tamo gdje se razmatraju i toplinski fenomeni fluida te gdje se događaju promjene unutarnje energije.

U izrazima (2-1) i (2-6) s lijeve strane jednadžbe nalaze se izrazi koji se često u literaturi zapisuju kao materijalna derivacija: promjene polja brzine i unutarnje energije su ovisne o vremenski i prostorno promjenjivom polju brzina unutar fluida. Definicija materijalne derivacije je:

$$\frac{D\varphi}{Dt} = \frac{\partial \varphi}{\partial t} + \vec{u} \cdot \nabla \varphi \tag{2-7}$$

Pa jednadžbe momenta i energije možemo zapisati kao:

$$\rho \frac{D\vec{u}}{Dt} = -\nabla p + \mu \nabla^2 \vec{u} + \vec{f}$$

$$\rho \frac{D\varepsilon}{Dt} = \nabla \cdot (K_h \nabla T) - p \nabla \cdot \vec{u}$$

2.3 Eulerove jednadžbe

Pojednostavljenje Navier-Stokesovih jednadžbi je moguće pretpostavljanjem neviskoznog toka. Iako je viskoznost fluida jedno od temeljnih svojstava, u nekim se situacijama može izostaviti i tada se primjenjuju Eulerove jednadžbe.

Jednadžba očuvanja mase ostaje istovjetna (2-2) odnosno (2-4) za nestlačivi tok.

Jednadžba gibanja dobiva se iz (2-1) izbacivanjem izraza za viskoznost:

$$\rho\left(\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u}\right) = -\nabla p + \vec{f} \tag{2-8}$$

Neviskozan tok ima u jednadžbi energije posljedicu nestajanja izraza za toplinsku vodljivost. Naime viskoznost je efekt trenja unutar fluida, a trenje se manifestira prijenosom toplinske energije. Ako nema viskoznog ponašanja, nema ni utjecaja toplinske vodljivosti na energiju sustava.

Jednadžba energije prelazi iz (2-6) u:

$$\rho\left(\frac{\partial \varepsilon}{\partial t} + \vec{u} \cdot \nabla \varepsilon\right) = -p\nabla \cdot \vec{u} \qquad (2\text{-}9)$$

2.4 Podjela prostora u simulacijama

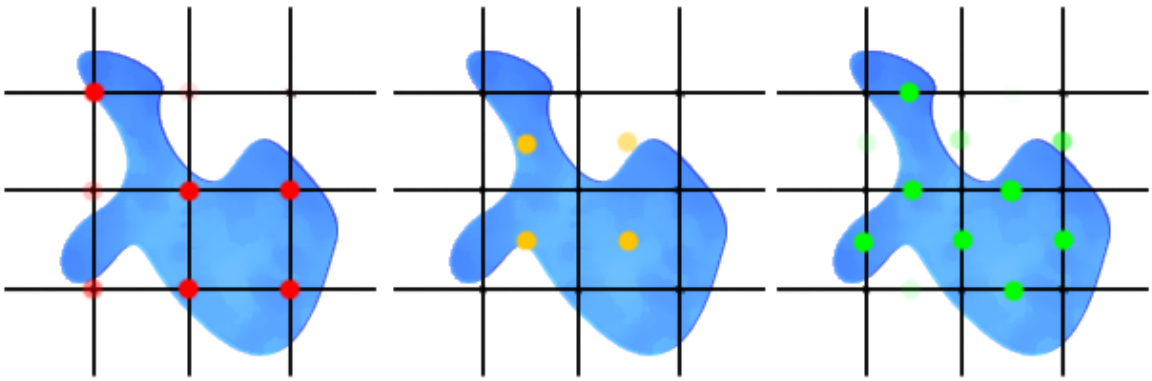
Kao što je ranije spomenuto, u simulacijama postoje dva načina podjele volumena unutar kojeg se fluid nalazi i unutar kojeg se vrši postupak rješavanja jednadžbi i integracije. Jedan način je „uzorkovanjem“ prostora fluida u pravilno raspoređenim točkama (prostorna rešetka). Drugi je način oblikovanjem fluida kao skupa nedjeljivih, nespojivih, nepromjenjivih čestica koje predstavljaju masu fluida i ponašaju se pod utjecajem drugih čestica.

Eulerova rešetka

Prostor unutar kojeg se fluid nalazi i unutar kojeg se mijenja moguće je diskretizirati i prikazati prostornom rešetkom. Moguća mjesta promatranja (tj. uzorkovanja) fluida su:

- presjecište linija rešetke (crveno)
- središte ćelija unutar rešetke (narančasto)
- središte između dvaju presjecišta (zeleno)

Slika 2.1 prikazuje moguća mjesta uzorkovanja, a intenzitetima boja prikazana je ovisnost o postojanju fluida.



Slika 2.1 Rešetka fluida s intenzitetima ovisnima o postojanju fluida

Kako bi se pravilno izračunale derivacije za rješavanje jednadžbi fluida, efikasnije je uzorkovati svojstva na različitim mjestima. Primjer takvog rješenja i njegove učinkovitosti

je rešetka markera i ćelija (*MAC, engl. Marker and cell*) u kojoj se vrijednosti tlaka i gustoće pamte u središtima ćelija, a brzine na rubovima ćelija (Enright, 2002.).

Prednost rešetke je praktičnost, lakoća implementacije i algoritmi poput semi-Lagrangian metode koji osiguravaju bezuvjetnu stabilnost. Za vizualizaciju ovakvog fluida postoji nekoliko interesantnih i efikasnih metoda kao što su razinski skupovi, a moguće je koristiti i neke metode pogodne za čestične sustave, kao na primjer pokretne kocke. Dodatna prednost je jednostavna mogućnost paralelizacije na GPU.

Glavni nedostatak ove metode je potreba pamćenja podataka za čitavu prostornu rešetku pa tako prostorna i vremenska složenost iznose$O(n^k)$ gdje je k broj prostornih dimenzija. Za potrebe malo većih i preciznijih simulacija (red veličine $n = 256,\ k = 3$) postizanje interaktivne simulacije nije lako. Drugi nedostatak je problem s očuvanjem mase koji nastaje kada jednadžba (2-4) ne vrijedi, a rješenje tog problema su metode koje kombiniraju prostornu rešetku i čestice. Iako popravlja problem nestajanja mase, to rješenje uvodi dodatnu složenost.

Lagrangianov sustav čestica

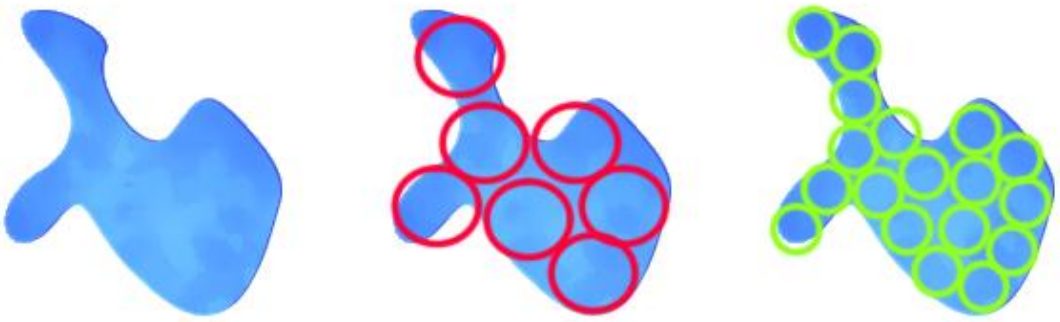
Svijet, tvari koje nas okružuju i sustavi u kojima postojimo sastavljene su od subatomskih čestica, od atoma, odnosno od molekula. Fluidi su ništa više nego nakupine molekula čije ponašanje i međusobna interakcija podliježu zakonima fizike fluida. Stoga je ideja ovog pristupa modelirati fluid kao grupe čestica koje djeluju silama fluida jedna na drugu.

Tipičan fluid sastavljen je od velikog broja čestica, reda veličine vrijednosti jednog mola $6{,}022045 \times 10^{23}$. Tako na primjer kilogram vode, čija je molarna masa $M_{H_2O} = 18{,}01528\,\frac{g}{mol}$, sadrži:

$$\frac{1}{M_{H_2O}} * 1000g = \frac{1000}{18{,}01528} = 55{,}5084\,mol = 3{,}3427 \times 10^{25}$$

čestica. U današnje vrijeme toliki broj čestica je računalno neizvediv te se fluidi aproksimiraju grupama čestica koje predstavljaju količinu fluida određene mase ili volumena. Na primjeru slike 2.2. su dvije takve grupacije, manjim i većim česticama. Tipično se u simulacijama koristi od 10^3 do 10^6 čestica. Simulacije u stvarnom vremenu su moguće do 75,000 čestica uz 15 sličica po sekundi (*fps, engl. frames per second*) kao

što su postigli (Goswami, 2010) ili do 65,000 čestica uz 16fps i ubrzanje u odnosu na CPU od 17 puta za taj broj čestica u (Harada, 2007.).



Slika 2.2 Čestice različitih veličina predstavljaju fluid

U čestičnim metodama nije lako dobiti derivacije svojstava koje su potrebne za rješavanje jednadžbi fluida te se one aproksimiraju na razne načine. Neka se ponašanja fluida mogu dodatno modelirati kao tipični sustav čestica i opruga što rezultira fizikalno netočnim, ali vizualno uvjerljivim simulacijama. U samim česticama pohranjuju se podatci o masi, gustoći, tlaku, temperaturi, volumenu uz neke dodatne podatke poput onih o susjedstvu ili o detekciji površinskih čestica.

Prednost ovih metoda je što je moguće uz mali broj čestica ostvariti uvjerljive rezultate pri interaktivnim brzinama iscrtavanja. Iako je vremenska složenost uobičajenih algoritama $O(n^2)$ zbog potrebe usporedbe svake čestice sa svakom drugom, moguće ju je smanjiti na $O(nm)$ gdje je m prosječan broj susjeda unutar rešetke susjedstva. Iako uvodimo prostornu rešetku, ona nije tako velikih dimenzija kao u Eulerovom pristupu te njeno korištenje povećava efikasnost metode. Još jedna bitna prednost korištenja čestica je i podrazumijevano očuvanje mase fluida, osim ako se čestice ne uklone ručno, one neće nestati iz sustava.

Nedostatak Lagrangian metoda je veća složenost izračuna zbog traženja susjednih čestica ili održavanja rešetke susjedstva. Postoje i poteškoće s vizualizacijom koja se svodi na metode traženja izopovršine kao što su pokretne kocke ili metakugle, ali postoje metode koje za određene situacije omogućuju učinkovit i brz prikaz. Više o vizualizaciji čestičnih fluida u poglavlju 4. Još jedan nedostatak su i poteškoće pri paralelizaciji postupaka zbog nepravilnog razmještaja čestica u prostoru.

3 Hidrodinamika sustava izglađenih čestica

U Lagrangian metodama čestice fluida nose svojstva fluida: skalarne vrijednosti koje predstavljaju masu, gustoću, tlak i vektorske vrijednosti brzine, akceleracije i sile na fluid. SPH metoda interpolira interesantna svojstva u nekoj točki prostora na temelju čestica – nosača svojstava koje se nalaze u blizini te točke prostora. To lokalno okruženje unutar kojeg se čestice uzimaju u obzir naziva se duljina izglađivanja (*engl. smoothing length*) jer se unutar tog volumena prostora svojstva izglađuju koristeći neku jezgrenu funkciju. Za svaki korak integracije, svojstva čestica se interpoliraju na temelju okolnih čestica koristeći upravo ovaj princip. Ovaj postupak je opravdan jer u bilo kojoj točci prostora može izraziti neko svojstvo i time zamjenjuje potrebu za prostornom rešetkom. Derivacije svojstava koje su potrebne u mnogim jednadžbama se jednostavno izražavaju i stoga je ovaj postupak pogodan za primjenu u mnogim fizikalnim sustavima.

3.1 Izglađivanje svojstava

Kako bi se izračunala vrijednost nekog skalarnog svojstva neke čestice, SPH metoda to svojstvo interpolira kao težinsku sumu utjecaja okolnih čestica izglađenih nekom jezgrom. Općenito, za neku točku prostora $\vec{r}$ osnovni SPH izraz za izračun svojstva A izveden je u (Monaghan, 1992.) i glasi:

$$A(\vec{r}) = \int A(\vec{r}')\,\delta(|\vec{r} - \vec{r}'|)d\vec{r}' \tag{3-1}$$

gdje je δ Diracova delta funkcija. Prema SPH metodi svojstva se izglađuju jezgrenim funkcijama pa se Dirac funkcija zamjenjuje općenitom jezgrom koja ima svojstvo:

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

i za tu jezgru vrijedi:

$$\int W(\vec{r} - \vec{r}', h)\, d\vec{r}' = 1 \tag{3-2}$$

kao i za Diracovu delta funkciju. Uvrštavanjem u (3-1) dobiva se:

$$A(\vec{r}) = \int A(\vec{r}')\, W(|\vec{r} - \vec{r}'|, h)d\vec{r}' \tag{3-3}$$

Točke prostora $\vec{r}'$ se diskretiziraju nad skupom$N$ čestica fluida i integral iz (3-3) prelazi u izraz za sumu:

$$A(\vec{r}) = \sum_{j=0}^{N-1} \frac{m_j}{\rho_j} A_j\, W\left(|\vec{r} - \vec{r}_j|, h\right) \tag{3-4}$$

U (3-4) omjer $\frac{m}{\rho}$ je prostorni utjecaj (tj. volumen utjecaja) čestice, dobiven iz jednadžbe za gustoću.

Pošto se svojstva računaju na pozicijama čestica, $A(\vec{r})$ će uvijek biti računat za neku česticu i s pozicijom $\vec{r}_\mathrm{i}$, tako da (3-4) u praksi postaje:

$$A_\mathrm{i} = \sum_{\mathrm{j}\neq\mathrm{i}} \frac{m_j}{\rho_j} A_j \, W\left(|\vec{r}_\mathrm{i} - \vec{r}_\mathrm{j}|, h\right) \tag{3-5}$$

Gradijent svojstva je sada naprosto:

$$\nabla A_\mathrm{i} = \sum_{\mathrm{j}\neq\mathrm{i}} \frac{m_j}{\rho_j} A_j \, \nabla W\left(|\vec{r}_\mathrm{i} - \vec{r}_\mathrm{j}|, h\right) \tag{3-6}$$

Laplasijan svojstva je također jednostavan:

$$\nabla^2 A_\mathrm{i} = \sum_{\mathrm{j}\neq\mathrm{i}} \frac{m_j}{\rho_j} A_j \, \nabla^2 W\left(|\vec{r}_\mathrm{i} - \vec{r}_\mathrm{j}|, h\right) \tag{3-7}$$

Ovo je jedna od bitnih prednosti SPH metode, što se gradijent i Laplasijan vrlo jednostavno računaju, potrebno je samo poznavati gradijent i Laplasijan jezgrene funkcije. O primjeni izglađenih svojstava više u sljedećem potpoglavlju.

3.2 Fizika fluida u SPH sustavu

Prema jednadžbi (2-1) sile koje djeluju na fluid su sile tlaka, sile viskoznosti i vanjske sile. Jednostavnije, Navier-Stokesovu jednadžbu momenta možemo zapisati kao sustav:

$$\rho\vec{a} = \vec{f}_{tlak} + \vec{f}_{viskoznost} + \vec{f}_{vanjska} \tag{3-8}$$

$$\vec{f}_{tlak} = -\nabla p \tag{3-9}$$

$$\vec{f}_{viskoznost} = \mu\nabla^2\vec{\mathrm{u}} \tag{3-10}$$

U jednadžbama (3-9) i (3-10) javljaju se gradijent tlaka i Laplasijan brzine čestice, a te diferencijale možemo izraziti kao izglađena svojstva.

Vrijednost tlaka

Prije računanja sile tlaka, potrebno je za svaku česticu izračunati iznos tlaka točke prostora gdje se čestica nalazi. To je moguće korištenjem jednadžbe idealnog plina:

$$p = k\rho$$

gdje je k konstanta plina ovisna o temperaturi.

Na česticu djeluje to veći tlak što je više čestica u njenom okruženju i ona tada ima težnju izjednačavanja gustoće u smjeru područja manje gustoće, dok se gustoća okruženja čestice ne približi gustoći fluida. U slučaju područja manje gustoće ponašanje je isto. Tlak se javlja kao posljedica razlike gustoće u odnosu na gustoću mirovanja ρ_0, pa je tlak efikasnije računati kao:

$$p_i = k(\rho_i - \rho_0) \tag{3-11}$$

Za izračun tlaka potrebno je izračunati gustoću čestice pa primjenom (3-5) na vrijednost gustoće slijedi:

$$\rho_\mathrm{i} = \sum_{\mathrm{j}\neq\mathrm{i}} \frac{m_j}{\rho_j} \rho_j \, W\left(|\vec{r}_\mathrm{i} - \vec{r}_\mathrm{j}|, h\right)$$

$$\rho_\mathrm{i} = \sum_{\mathrm{j}\neq\mathrm{i}} \mathrm{m_j} \, W\left(|\vec{r}_\mathrm{i} - \vec{r}_\mathrm{j}|, h\right) \tag{3-12}$$

Sila tlaka

Gradijent tlaka za česticu i primjenom gradijenta izglađenog svojstva (3-6) postaje:

$$\nabla \mathrm{p_i} = \sum_{\mathrm{j}\neq\mathrm{i}} \frac{m_j}{\rho_j} \mathrm{p}_j \, \nabla W\left(|\vec{r}_\mathrm{i} - \vec{r}_\mathrm{j}|, h\right)$$

pa je izraz za silu tlaka na česticu i:

$$\vec{f}_\mathrm{i}^{\mathrm{tlak}} = -\sum_{\mathrm{j}\neq\mathrm{i}} \frac{m_j}{\rho_j} p_j \, \nabla W\left(|\vec{r}_\mathrm{i} - \vec{r}_\mathrm{j}|, h\right) \tag{3-13}$$

Problem s ovim pristupom je što za dvije čestice i i j sile nisu simetrične i treći Newtnov zakon sile akcije i sile reakcije ne vrijedi. Iako imaju istu vrijednost gradijenta jezgre, u općenitom slučaju ne vrijedi $\frac{m_i}{\rho_i} p_\mathrm{i} = \frac{m_j}{\rho_j} p_j$ što stvara nesimetričnost sila.

Primjer rješenja je računanje simetričnih sila kao što je opisao (Desbrun, 1996.):

$$\vec{f}_\mathrm{i}^{\mathrm{tlak}} = -\mathrm{m_i} \sum_{\mathrm{j}\neq\mathrm{i}} \mathrm{m_j} \left(\frac{\mathrm{p_i}}{\mathrm{\rho_i^2}} + \frac{\mathrm{p_j}}{\mathrm{\rho_j^2}}\right) \nabla W\left(|\vec{r}_\mathrm{i} - \vec{r}_\mathrm{j}|, h\right) \tag{3-14}$$

Moguće drugo rješenje koje odgovara zahtjevima brzine i stabilnosti predložio je (Müller, 2003.):

$$\vec{f}_i^{tlak} = -\sum_{j\neq i} m_j \frac{p_i + p_j}{2\rho_j} \nabla W\left(|\vec{r}_i - \vec{r}_j|, h\right) \tag{3-15}$$

U većini radova vezanih uz SPH simulaciju fluida, koristi se (3-15).

Sila viskoznosti

Izraz sile viskoznosti (3-10) za česticu i primjenom Laplasijanaizglađenog svojstva prelazi u:

$$\vec{f}_{\mathrm{i}}^{\mathrm{viskoznost}} = \mu \sum_{\mathrm{j}\neq\mathrm{i}} \frac{m_j}{\rho_j} \mathrm{v}_j \nabla^2 W\left(|\vec{r}_{\mathrm{i}} - \vec{r}_{\mathrm{j}}|, h\right) \tag{3-16}$$

Kao i kod tlaka, nastaje problem s trećim Newtnovim zakonom. Rješenje je umjesto brzine susjedne čestice koristiti razliku brzina. Taj se postupak opravdava težnjom čestica za ujednačavanjem brzina, što znači da će velika razlika brzina stvoriti veliku silu koja se odupire velikim razlikama brzina za viskozan fluid (Müller, 2003.). Konačan izraz je:

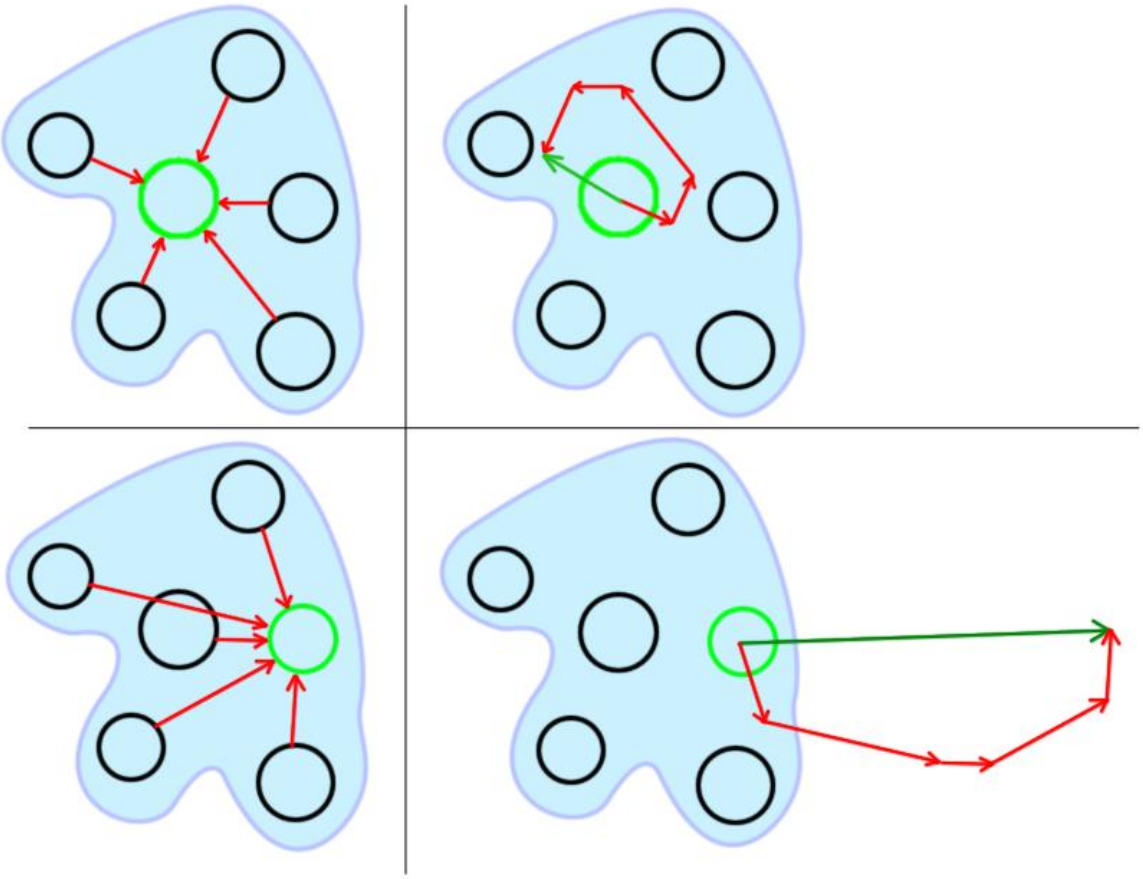
$$\vec{f}_{\mathrm{i}}^{\mathrm{viskoznost}} = \mu \sum_{\mathrm{j}\neq\mathrm{i}} \frac{m_j}{\rho_j} (\mathrm{v}_j - \mathrm{v}_{\mathrm{i}}) \nabla^2 W\left(|\vec{r}_{\mathrm{i}} - \vec{r}_{\mathrm{j}}|, h\right) \tag{3-17}$$

3.3 Površinska napetost

Sila površinske napetosti nije dio Navier-Stokesovih jednadžbi (2-1) i potrebno ih je dodatno modelirati. Razlog zašto napetost nije eksplicitno navedena je zato što su sile napetosti prisutne na prijelazu iz jednog fluida u drugi, a jednadžbe opisuju ponašanje samo jednog fluida. U SPH sustavu se najčešće radi s jednim fluidom i bez sila napetosti rub fluida bio bi nestabilniji zbog razlika u gustoćama na rubu. Naime u središtu fluida sile se izjednačavaju i poništavaju jer su čestice okružene, dok se na rubovima javljaju sile tlaka koje djeluju u smjeru suprotnom od volumena fluida, u smjeru gdje nema drugih čestica.

Morris (2000.) je primijenio SPH metodu na modeliranje isključivo sila površinske napetosti i većina SPH simulacija koristi njegov postupak za modeliranje tih sila. S obzirom na to da sile napetosti djeluju tako da smanje zakrivljenost površine, one su najjače tamo gdje je zakrivljenost površine fluida najveća. Kako bi se pronašao rub fluida uvodi se svojstvo fluida zvano polje boja (*engl. color field*) koje daje mjeru okruženosti drugim česticama. Vrijednost polja boja je 1 tamo gdje se čestica nalazi, a 0 inače. To znači da će izglađeno polje boja imati gradijent manje duljine za čestice koje su u sredini fluida, dok će na rubnim česticama duljina gradijenta biti veća. Na slici 3.1. dan je primjer gradijenta polja boja unutar fluida (gornji dio slike) te gradijenta polja boja na rubu fluida (donji dio slike). Sila napetosti se javlja tamo gdje je polje boja, odnosno duljina gradijenta polja

boja, veća od unaprijed definiranog praga, a djeluje u smjeru gradijenta polja boja koji ujedno predstavlja i normalu na površinu fluida. Normala je okrenuta izvan volumena fluida kao što to prikazuje tamno zelena strelica na donjem djelu slike 3.1.



Slika 3.1 Primjer polja boje u središtu fluida (gore) i na rubu fluida (dole)

Samo polje boja računa se kao izglađeno svojstvo postojanja čestice (vrijednost 1 tamo gdje se čestica nalazi, 0 tamo gdje nema čestica):

$$c_i = \sum_{j \neq i} \frac{m_j}{\rho_j} W\left(|\vec{r}_i - \vec{r}_j|, h\right) \tag{3-18}$$

Normala površine fluida je gradijent polja boja:

$$\vec{n} = \nabla c_i = \sum_{j \neq i} \frac{m_j}{\rho_j} \nabla W\left(|\vec{r}_i - \vec{r}_j|, h\right) \tag{3-19}$$

Zakrivljenost površine određuje gdje treba primijeniti silu napetosti, a računa se kao Laplasijana polja boja:

$$\kappa = -\frac{\nabla^2 c_i}{|\vec{n}|} \tag{3-20}$$

Sila napetosti je sada:

$$\vec{f}_i^{napetost} = \sigma\kappa\vec{n} \tag{3-21}$$

gdje je σ koeficijent površinske napetosti

Sila napetosti dodaje se silama uz sile (3-9) i (3-10) onda kada je zadovoljen uvjet da gradijent prelazi unaprijed definirani prag:

$$|\vec{n}| > c_{granica}$$

Iako je ovaj model napetosti popularan, postoje bolja, egzaktnija, ali ujedno i složenija rješenja.

3.4 Jezgrene funkcije

S ciljem bolje aproksimacije ponašanja pojedinih sila u fluidu, osmišljene su posebne jezgre za razne situacije. Korištenje prikladne jezgre jedan je od osnovnih uvjeta ispravne, stabilne i brze simulacije SPH fluida. Postoji nekoliko uvjeta koji jezgru čine boljom:

- parnost jezgre: $f(x) = f(-x)$
- normaliziranost jezgre: jednadžba (3-2)
- kontinuiranost druge derivacije
- kompaktnost: vrijednost jezgre je 0 za $r < h$, gdje je $r$duljina vektora udaljenosti $\vec{r}$, a h je duljina izglađivanja

Parne i normalizirane jezgre imaju greške interpolacije drugog reda $O(h^2)$. Kontinuiranost druge derivacije je bitna jer takva jezgra umanjuje greške koje nastaju zbog aproksimacije integrala sumacijom (Monaghan, 1992.). Kompaktnost smanjuje složenost računanja, jer možemo zanemariti čestice koje su udaljenije od duljine izglađivanja.

Nedostatak korištenja samo jedne jezgre je to što nije prilagođena specifičnostima pojedinih sila. Promjenu je uveo Müller (2003.) koji je definirao tri jezgre koje se redovito javljaju u SPH simulacijama, a posebno su osmišljene kako bi bolje odgovarale aproksimaciji sila unutar fluida. Postoje posebne jezgre za računanje sila viskoznosti i sila tlaka, te još jedna jezgra za općenitu upotrebu. Sve su jezgre parne, normalizirane i kontinuiranih drugih derivacija. Dodatno, definirane su na području $0 \leq r \leq h$ tj. za sve vrijedi:

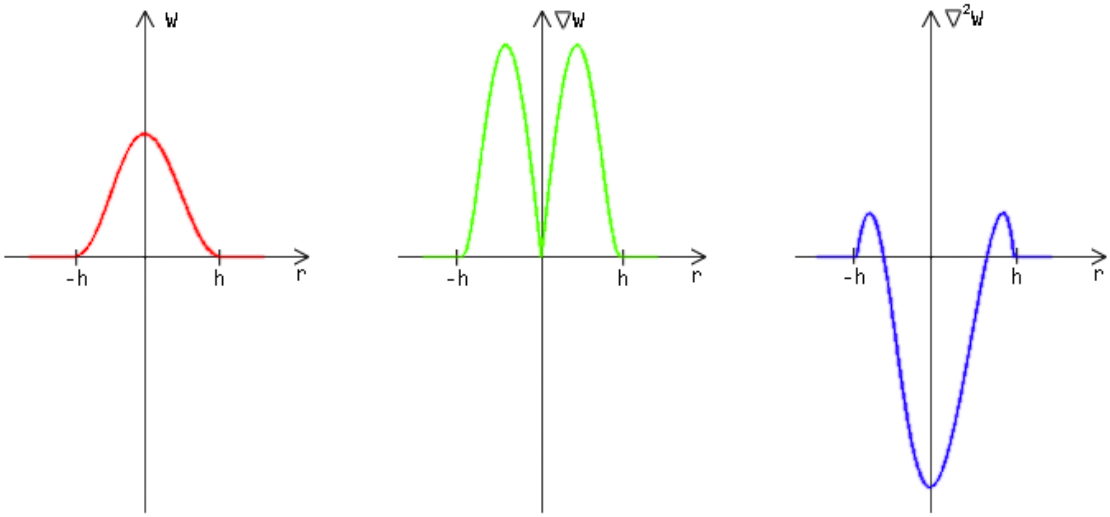
$$W(\vec{r},h) = 0, r > h \tag{3-22}$$

Na slici 3.2. nalaze se grafovi prve jezgre W_{poly6} koja je polinom šestog stupnja. Graf funkcije je crvene boje, graf gradijenta je zelene boje, a Laplasijan je plave boje. Svi su grafovi simetrični zbog svojstva parnosti jezgre. Slijede jednadžbe te jezgre:

$$W_{poly6}(\vec{r},h) = \frac{315}{64\pi h^9}(h^2 - r^2)^3$$

$$\nabla W_{poly6}(\vec{r},h) = -\vec{\mathrm{r}}\frac{945}{32\pi h^9}(h^2 - r^2)^2$$

$$\nabla^2 W_{poly6}(\vec{r},h) = \frac{945}{8\pi h^9}(h^2 - r^2)(\mathrm{r}^2 - \frac{3}{4}(h^2 - r^2))$$



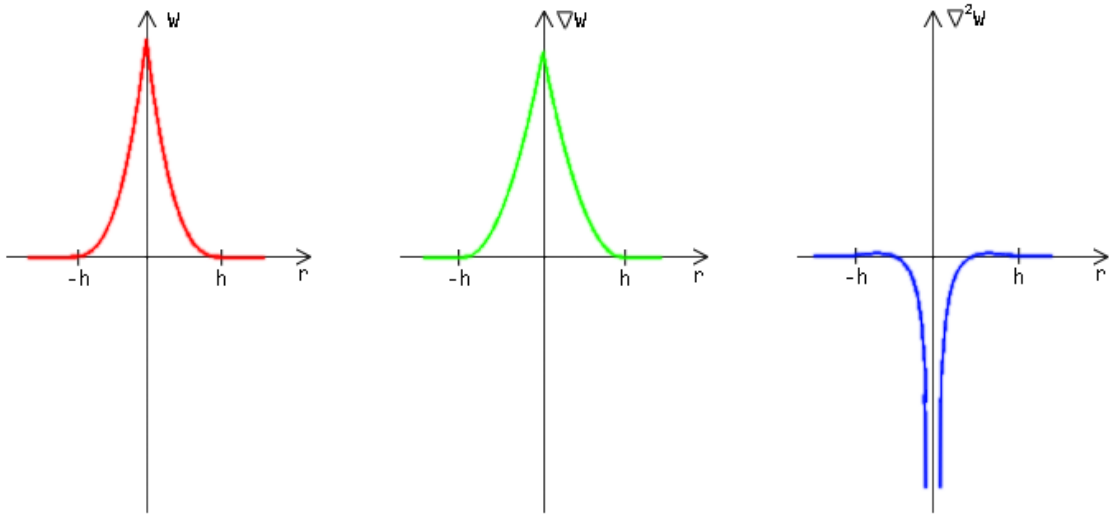
Slika 3.2 Graf *poly6* jezgre, njena gradijenta i Laplasijana

Sljedeća je jezgra W_{spiky} korištena za izračun sila tlaka, a temeljena je na jezgri koju je osmislio Desburn (1996.). Ova jezgra ima dodatan zahtjev a taj je da njen gradijent ne nestaje kako se r približava nuli. Ako taj zahtjev nije zadovoljen, na malim udaljenostima sila tlaka će iščeznuti i čestice će se grupirati u nakupine. Problem je što gradijent sadrži normalizirani vektor udaljenosti kojem se za udaljenosti bliske nuli javljaju nedefinirane vrijednosti. Na 3.3. nalaze se skalirani grafovi jezgre, njenog gradijenta (s ograničenjem za vrijednosti bliske nuli) i Laplasijana, a slijede njihove jednadžbe:

$$W_{spiky}(\vec{r},h) = \frac{15}{\pi h^6}(\mathrm{h} - \mathrm{r})^3$$

$$\nabla W_{spiky}(\vec{r},h) = -\hat{\mathrm{r}}\frac{45}{\pi h^6}(\mathrm{h} - \mathrm{r})^2$$

$$\nabla^2 W_{spiky}(\vec{r},h) = -\frac{90}{\pi \mathrm{h}^6}(\frac{\mathrm{h}^2}{\mathrm{r}} - 3\mathrm{h} + 2\mathrm{r})$$



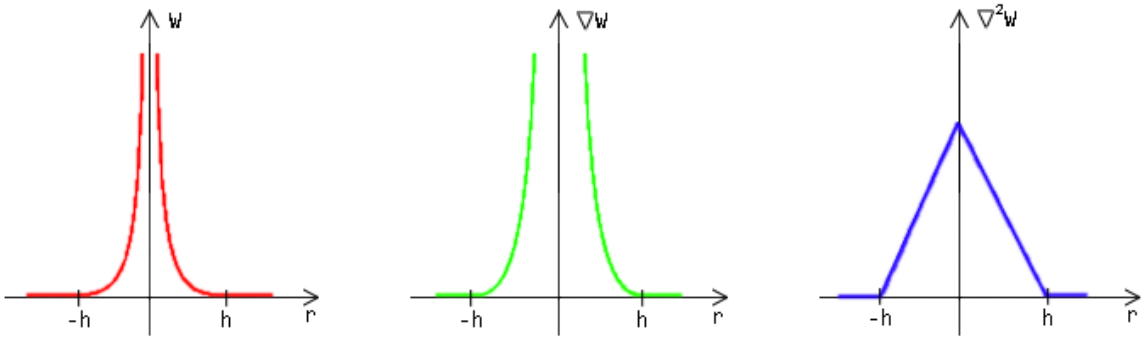
Slika 3.3 Graf *spiky* jezgre, njena gradijenta i Laplasijana

Posljednja jezgra $W_{viscosity}$ koristi se za izračun viskoznosti te je stoga bitan njen Laplasijan. Oblikovana je jezgra čiji je Laplasijan pozitivan i ravnomjerno izglađuje udaljenost. Ako se za viskoznost koristi jezgra s negativnim vrijednostima Laplasijana, umjesto viskoznog ujednačavanja brzina događat će se povećanja razlike brzina. Ova jezgra zbog svojih svojstava povećava stabilnost simulacije i uz nju nije potrebno raditi dodatno prigušivanje (Müller, 2003.). Na 3.4. nalaze se skalirani grafovi jezgre, njenog gradijenta i Laplasijana, a slijede njihove jednadžbe:

$$W_{viscosity}(\vec{r},h) = \frac{15}{2\pi h^3}(-\frac{\mathrm{r}^3}{2h^3} + \frac{r^2}{h^2} + \frac{h}{2r} - 1)$$

$$\nabla W_{viscosity}(\vec{r},h) = ?$$

$$\nabla^2 W_{viscosity}(\vec{r},h) = \frac{45}{\pi \mathrm{h}^6}(\mathrm{h} - \mathrm{r})$$



Slika 3.4 Graf *viscosity* jezgre, njena gradijenta i Laplasijana

Razlike između navedenih jezgri su velike te korištenjem isključivo jedne ili korištenjem krivih jezgara dolazi do problema stabilnosti i loših vizualnih rezultata.

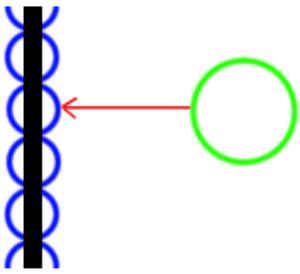
3.5 Kolizije i interakcija

Kolizija dvije čestice

U SPH sustavu nema sudara čestica u uobičajenom smislu jer čestice međusobno djeluju izglađenim poljem sila tlaka, viskoznosti i napetosti bez uzimanja u obzir fizičkih kolizija. Problem nastaje kada zbog velikih tlakova i neprimjerenog koraka integracije dvije čestice završe jedna blizu druge. Gradijent udaljenosti počinje težiti prema beskonačnosti (zbog dijeljenja nulom) što uzrokuje neželjeno ponašanje. Ako se takvo ponašanje spriječi ograničavanjem duljine gradijenta, pojavit će se drugi problem. S obzirom na to da je pozicija takvih čestica slična, sve izglađene vrijednosti će im biti slične, te će njihovo ponašanje konvergirati. Mogući su efekti stvaranja grupa čestica koje se ne mogu odvojiti jedna od druge, jer je gradijent ograničen. Više o tome u poglavlju 5.

Sila reakcije površine kolizije

U općenitom slučaju, fluid nije jedini objekt u sceni te je potrebno modelirati kolizije s raznim površinama. To primjerice mogu biti ravnine, cilindri, sfere, kvadri ili čitave poligonalne mreže. Uobičajen pristup je pretpostaviti da površine djeluju na fluid na isti način na koji čestice fluida međusobno djeluju, pretpostavlja se da su površine sačinjene od nepomičnih čestica fluida kao što to prikazuje slika 3.5.



Slika 3.5 Kolizija čestice i površine

Potrebno je računati sile tlaka i viskoznosti za sve površine unutar duljine izglađivanja čestica, a to je moguće koristeći (3-15) za tlak i (3-17) za viskoznost:

$$\vec{f}_{\mathrm{i}}^{\mathrm{tlakaKolizije}} = -\sum_{\mathrm{j}}^{\mathrm{N_{površina}}} \mathrm{m_j} \frac{\mathrm{p_i}+\mathrm{p_j}}{2\rho_{\mathrm{j}}} \nabla W\left(|\vec{r}_{\mathrm{i}} - \vec{r}_{\mathrm{j}}|, h\right)$$

$$\vec{f}_{\mathrm{i}}^{\mathrm{viskoznostiKolizije}} = \mu \sum_{\mathrm{j}}^{\mathrm{N_{površina}}} \frac{m_j}{\rho_j} (\mathrm{v}_j - \mathrm{v_i}) \nabla^2 W\left(|\vec{r}_{\mathrm{i}} - \vec{r}_{\mathrm{j}}|, h\right)$$

S obzirom na to da su te sile obično sile reakcije površine one su simetrične silama kojima čestica djeluje na površinu. Ako nisu dostupni podatci o masi djela površine s kojim

se čestica sudara i podatci o gustoći površine, rješenje je računati sile kao da se na površini nalazi čestica ista kao ona koja se sudara s površinom. Jednadžbe su tada:

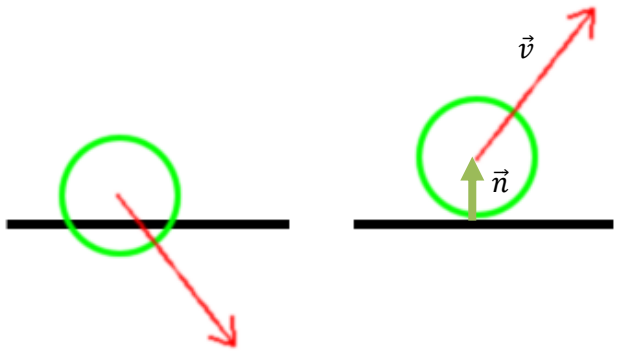
$$\vec{f}_{\mathrm{i}}^{\mathrm{tlakaKolizije}} = -\mathrm{m_i}\frac{\mathrm{p_i}}{\rho_{\mathrm{i}}}\sum_{\mathrm{j}}^{\mathrm{N_{površina}}}\nabla W\left(|\vec{r}_{\mathrm{i}} - \vec{r}_{\mathrm{j}}|, h\right) \tag{3-23}$$

$$\vec{f}_{\mathrm{i}}^{\mathrm{viskoznostiKolizije}} = \mu\frac{m_{\mathrm{i}}}{\rho_{\mathrm{i}}}\sum_{\mathrm{j}}^{\mathrm{N_{površina}}}(\mathrm{v}_j - \mathrm{v_i})\,\nabla^2 W\left(|\vec{r}_{\mathrm{i}} - \vec{r}_{\mathrm{j}}|, h\right) \tag{3-24}$$

Ako je površina nepomična, sile se računaju samo za česticu, a ako je površina pridružena pomičnom objektu, onda je pomoću (3-23) i (3-24) moguće dobiti i utjecaj fluida na objekt uronjen u fluid. Uz dovoljno detaljnu raspodjelu površina, moguće bi bilo ostvariti složene efekte poput uzgona i valjanja predmeta na površini fluida.

Neprobojna površina kolizije

Usprkos modeliranju sila na površine koje nastaju pri koliziji, moguće je da te sile nisu dovoljno snažne kako bi spriječile česticu u probijanju površine. Proboj u većini simulacija nije željeno ponašanje jer bi uzrokovao ulazak čestica fluida u objekt interakcije ili bježanje čestica fluida iz scene. Zato se takve situacije zasebno obrađuju izravnim sprječavanjem proboja. Čestica se odmiče od površine, a komponenta brzine okomita na površinu se zrcali (Müller, 2003.) kao što je predočeno na slici 3.6.



Slika 3.6 Trenutak kolizije s površinom i razrješenje sudara

Ako su ravnine poravnate s osima, onda je potrebno samo promijeniti predznak odgovarajuće komponente brzine. U općenitom slučaju potrebno je zrcaliti vektor brzine oko normale na površinu što se postiže jednadžbom:

$$\vec{v}_{zrcaljena} = \vec{v} - 2\vec{n}\frac{\vec{v}\cdot\vec{n}}{||\vec{n}||^2} \tag{3-25}$$

gdje je $\vec{n}$ normala površine ili u praksi jednostavnije vektor prema čestici.

3.6 Metode integracije

Stabilnost svih simulacija ovisi u velikoj mjeri o odabranoj metodi integracije. Naravno, korištenjem pretjerano složene metode postiže se suprotan efekt smanjenja performansi zbog složenog računa. S obzirom na to da se računaju samo pozicija i brzina, moguće je koristiti uobičajene metode poput leapfrog metode ili prediktor-korektor sheme uz koje je poželjno koristiti i korekciju vremenskog koraka (Monaghan, 1992.). U SPH animacija mase često koristi upravoleapfrog metoda kao na primjer u (Desbrun, 1996.).

Osnovne jednadžbe te metode primijenjene na integraciju pozicije i brzine su:

$$\vec{x}_i = \vec{x}_{i-1} + \vec{v}_{i-\frac{1}{2}}\Delta t$$

$$\vec{v}_{i+\frac{1}{2}} = \vec{v}_{i-\frac{1}{2}} + \vec{a}_i\Delta t$$

gdje je $\vec{a}_i$ rezultat djelovanja sila na česticu u tom koraku dobiven iz (3-8):

$$\vec{a}_i = \frac{\vec{f}^{ukupno}}{\rho} \tag{3-26}$$

U leapfrog metodi se pozicija računa na temelju brzine između dva koraka integracije. Na neki način vremenski koraci za izračun pozicije i brzine se preskaču (od tuda naziv metode, skok žabe). Sustav je moguće izraziti i pomoću cijelih indeksa te on tada postaje:

$$\vec{x}_{i+1} = \vec{x}_i + \vec{v}_i\Delta t + \vec{a}_i\frac{\Delta t^2}{2}$$

$$\vec{v}_{i+1} = \vec{v}_i + (\vec{a}_i + \vec{a}_{i+1})\frac{\Delta t}{2}$$

Razlika u odnosu na običnu Eulerovu metodu integracije je što se doprinos akceleracije na brzinu računa koristeći trenutnu i prethodnu vrijednost. Iako jako jednostavna, ova metoda je drugog reda i pogodna za SPH simulaciju te je stoga često prisutna u literaturi i radovima.

3.7 Vremenski korak

Za simulacije u stvarnom vremenu idealan vremenski korak jednak je ili veći od stvarnog protoka vremena kako bi tok simulacije mogao pratiti stvarno vrijeme animacije. Za potrebe znanstvenih simulacija nesklad vremenskog koraka simulacije i stvarnog vremenskog koraka animacije nije bitan što ne znači da je uvijek moguće odabrati jako mali Δt. Naime za većinu fizikalnih efekata potrebna je veća količina vremena kako bi se ti

efekti ostvarili i promotrili (na primjer pucanje brane ili sudar valova) pa manji koraci znače puno čekanja na kraj simulacije.

S druge strane, usprkos metodi integracije drugog reda i SPH sustavu koji je modeliran kako bi se umanjile greške (pomoću jezgri, poglavlje 3.4) preveliki vremenski korak uvodi nestabilnost i neželjeno ponašanje. Moguće rješenje je ograničavanje vremenskog koraka.

Za rješavanje parcijalnih diferencijalnih jednadžbi koje opisuju fluid numeričkim metodama temeljenim na metodi konačnih razlika postoji nužan uvjet za konvergenciju a zove se Courant-Friedrichs-Lewy uvjet ili kraćeCourant uvjet (Desbrun, 1996.). Taj uvjet glasi:

$$v\frac{\delta t}{\delta x} \leq 1$$

gdje je δt vremenski korak integracije, v je brzina, a δx je veličina rešetke. Uvjet ograničava promjenu gibanja fluida tako da se ne dozvoljava takav pomak koji bi uzrokovao preskakanje neke ćelije rešetke. U SPH metodi ne postoji rešetka pa se uvjet oblikuje tako da ne dozvoljava česticama da se međusobno prolaze bez utjecaja:

$$\delta t_i \leq \frac{h}{v_i} \tag{3-27}$$

gdje je v_i brzina čestice. Ukupni vremenski korak je najmanja vrijednost δt_i:

$$\delta t = \min_i \delta t_i$$

U (Desbrun, 1996.) spominje se mnogo kompliciraniji uvjet koji uzima u obzir brzinu zvuka u fluidu i koeficijent viskoznosti, a (3-27) je uzet kao pomoćni uvjet. Taj složeniji uvjet je:

$$\delta t_i \leq \delta t_i^{Courant} = \alpha \frac{h}{c + 0.6(c + 2\max_j \mu_{ij})}$$

gdje je c brzina zvuka u fluidu, α je Courantov broj, a μ_{ij} je koeficijent viskoznosti ovisan o udaljenosti i brzinama čestica. Iako je uporaba tog uvjeta uvriježena u astrofizici (Monaghan, 1992.) u animacijama se spominje kao dobra ideja, ali se češće koristi fiksni vremenski korak. Tako (Müller, 2003.) koristi korak od $0.001s$ dok (Yu, 2010.) koristi korak od $0.0001s$.

4 Metode vizualizacije fluida

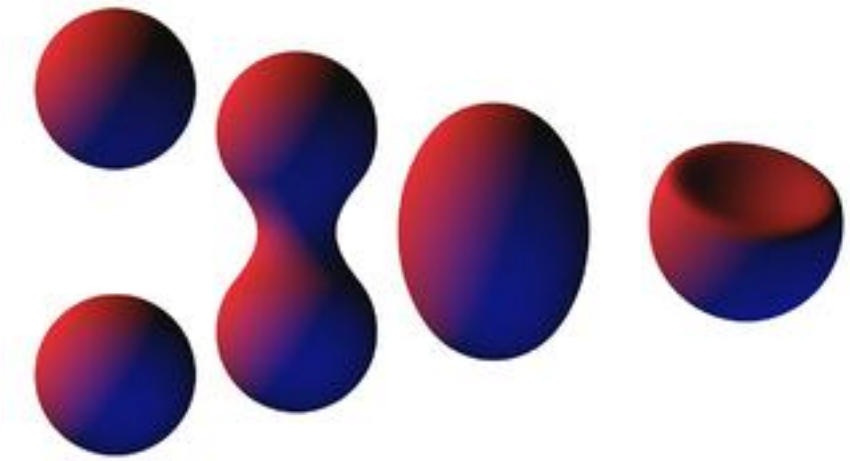
Za prikaz čestičnog fluida moguće je koristiti uobičajene metode za prikaz skupa čestica. Iako postoje mnoge metode ovisi o području primjene koja će metoda biti najinteresantnija i najpogodnija s obzirom na brzinu i efektivnost prikaza. Slijedi kratak opis nekoliko čestih metoda.

4.1 Metakugle

Metakugle su način vizualizacije izopovršine polja točaka koje se mogu opisati sferičnim objektima, a čiji se utjecaj na okolni prostor zbraja i time izglađuje. Općenita formula za metakugle je:

$$\sum_{i=0}^{n} f_i(x, y, z) \leq p$$

gdje je f_i jezgrena funkcija metakugle za točku prostora (x, y, z), a p je prag sume kojim se određuje veličina metakugli. Jezgra može biti formula za sferu, polinomijalna funkcija ili Gaussova krivulja. Na slici 4.1. nalaze se metakugle čiji se utjecaj zbraja i oduzima (zadnji primjer).



Slika 4.1 Metakugle (Glyde, 2007.)

Metakugle je moguće prikazati na više načina, a najčešći su postupak bacanja zrake i algoritam pokretnih kocki. Algoritam pokretnih kocki je interesantniji za slučaj simulacije u stvarnom vremenu pa je njegov opis izdvojen u posebnom potpoglavlju.

Bacanje zrake

Postupak bacanja zrake (*engl. Ray casting*) sastoji se od stvaranja i praćenja pravaca – zraka iz položaja oka za svaki fragment ekrana koji se prikazuje, traženja

presjecišta tih zraka s objektima na sceni i računanja boje za fragment kroz koji zraka prolazi. Princip ovog algoritma moguće je primijeniti i u drugim metodama vizualizacije fluida, ali je najisplativiji upravo za prikaz metakugli.

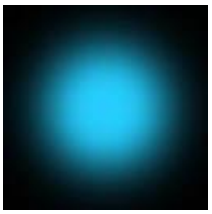
Postoji mnogo načina i varijacija algoritma za primjenu na metakuglama, a interesantan je rad (Kanamori, 2008.) koji prikazuje velik broj metakugli uz interaktivno vrijeme prikaza. Koriste metodu guljenja dubina (*engl. depth peeling*) koja se koristi za prikaz prozirnih objekata te algoritam Bézierovog odsjecanja (*engl. Bézier clipping*) za pronalazak presjecišta zrake i izopovršine metakugli. Algoritam su prilagodili izvođenju na GPU te postižu ubrzanja od 5 do 6 puta u odnosu na CPU inačicu. Na slici 4.2. nalazi se prikaz fluida njihovim algoritmom. Fluid je sastavljen od oko $100{,}000$ metakugli, a brzina animacije je $2.3\ fps$.

Slika 4.2 Fluid prikazan metakuglama [16]

4.2 Polje sličica

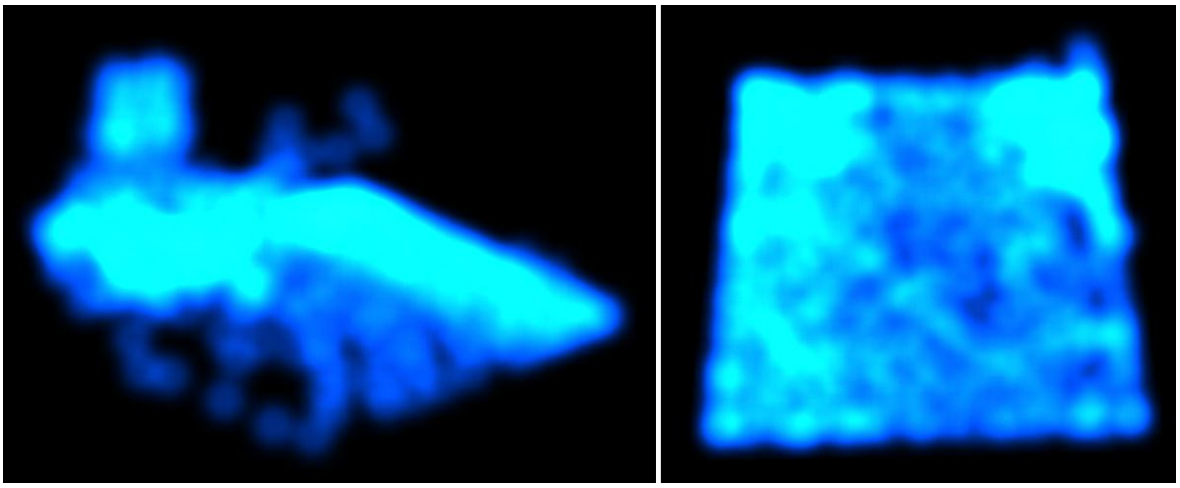
Ovo je najjednostavnija metoda za prikaz polja čestica, a sastoji se od crtanja četverokuta koji se nalazi na poziciji čestice. Ako se taj četverokut okrene prema promatraču, tada je riječ o tehnici oglasne ploče. Moguće je postići lijepe efekte i izglađen izgled fluida odabirom primjerene teksture za četverokut (primjerice kao na slici 4.3.) ili korištenjem posebnih metoda za prikaz.

Polje sličica moguće je realizirati na više načina. Najjednostavnije i najsporije je korištenje snage procesora kako bi se izračunale točke svih četverokuta okrenute prema oku i standardnog teksturiranog prikaza tih četverokuta.



Slika 4.3 Sličica za prikaz čestice

Drugi način je korištenjem gotovih funkcionalnosti koje pruža aplikacijsko programsko sučelje. Primjerice OpenGL ima podršku za prikaz polja sličica (*engl. Point sprites*) korištenjem gotovih funkcija, a na slici 4.4. nalazi se rezultat upravo takvog prikaza korištenjem teksture na slici 4.3.



Slika 4.4 Prikaz fluida pomoću ugrađenih OpenGL funkcionalnosti

Najsloženiji način prikaza polja sličica je korištenjem snage GPU i sjenčanja geometrije. U grafički protočni sustav šalju se samo pozicije čestica fluida, a četverokuti se generiraju sjenčanjem geometrije. Prednosti ove metode je mogućnost ostvarenja dodatnih efekata u programu za sjenčanje fragmenata. Nedostatak ovog pristupa je što zahtjeva računalo s grafičkom karticom nove generacije koja podržava sjenčanje geometrije (odnosno podržava DirectX 10 ili OpenGL 3.2), a ubrzanje u odnosu na ugrađenu funkcionalnost nije značajno.

Više o polju sličica izvedenom na GPU u poglavlju 5.5.

4.3 Pokretne kocke

Pokretne kocke su algoritam koji stvara poligonalni model površine stalne gustoće na temelju 3D podataka. Korištenjem pristupa podijeli pa vladaj za izradu unutarnje povezanosti elemenata prostora, stvara se tabela slučajeva koja određuje topologiju

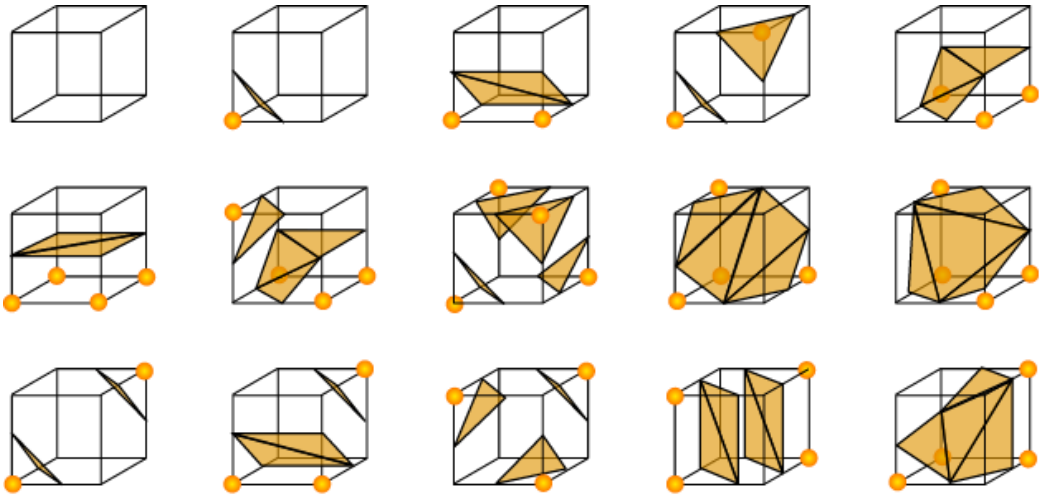
trokuta. Algoritam procesuira 3D podatke i računa vrhove trokuta koristeći linearnu interpolaciju. (Lorensen, 1987.)

Algoritam je osmišljen za vizualizaciju podataka nastalih prilikom medicinskog skeniranja. Riječ je o podatcima – brojevima smještenima u pravilnu 3D rešetku koji označavaju prisustvo/odsustvo materije. Od svojih početaka algoritam privlači mnogo pažnje i primjene, a posebice nakon godine 2005. kada je istekao njegov patent. Zbog svoje primjene u simulaciji fluida, u ovom se potpoglavlju posvećuje više pažnje opisu rada algoritma.

Općenito govoreći, algoritam stvara mrežu trokuta koja predstavlja izopovršinu opisanu pravilnom prostornom rešetkom nekog skalarnog polja. U slučaju fluida, prostorna će rešetka sadržavati vrijednosti koje označavaju postoji li u blizini neka čestica fluida ili ne. U centru čestice, rešetka će imati vrijednost 1.0, na rubovima čestice rešetka će poprimiti vrijednost ovisnu o udaljenosti od ruba omeđujućesfere čestice, dok će točke rešetke udaljene od ruba čestice poprimiti vrijednost 0.0.

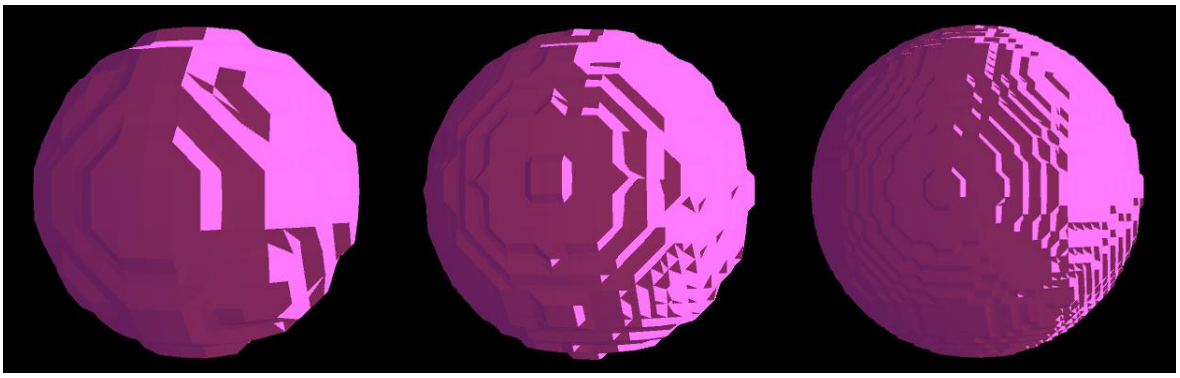
Ćelije prostorne rešetke nalaze se između osam susjednih točaka prostorne rešetke. Na temelju vrijednosti praga nalaze se točke rešetke koje ulaze u algoritam. Upotrebom tablice bridova pronalaze se točke interpolacije koje će biti vrhovi generiranih trokuta. Upotrebom tablice trokuta određeni su svi mogući slučajevi trokuta koje je potrebno generirati za određeni slučaj ćelije. Kako postoji osam vrhova ćelije, a moguće je isključiti/uključiti bilo koju od njih, postoji $2^8 = 256$ različitih konfiguracija kocke. Zbog simetričnosti i refleksije problem pokretnih kocki svodi se na 15 jedinstvenih slučajeva prikazanih na slici 4.5.

Tablica bridova sadrži 256 vrijednosti kojima se za svaki slučaj kocke određuju pozicije bridova. Svaki izlazni brid je kodiran kao jedan bit vrijednosti u tablici. Najniže bitovne pozicije (od 0 do 3) sadrže prednja četiri brida, srednje pozicije bitova (od 4 do 7) sadrže bridove stražnje plohe dok najviši bitovi (od 8 do 11) sadrže bridove u sredini kocke. Ako je bit na odgovarajućoj poziciji vrijednosti 1, pripadajući brid će biti korišten za izradu izlaznog trokuta. U tablici trokuta nalaze se redni brojevi bridova koji čine izlazne trokute za svaki od 256 slučajeva. Najgori slučaj generira 5 trokuta koji su određeni s 15 vrhova što znači da ta tablica ima dimenzije 256×16. U retku tablice trokuta koji odgovara slučaju kocke, svaka tri uzastopna broja različita od -1 određuju bridove izlaznog trokuta.



Slika 4.5 Slučajevi konfiguracije trokuta algoritma pokretnih kocki [32]

Na slici 4.6 nalazi se prikaz tri sfere, svaka je nacrtana koristeći pokretne kocke, ali uz različite veličine rešetke. Veća rešetka je potrebna za preciznije rezultate, ali ona također zahtijeva više računalne snage.



Slika 4.6 Sfera nacrtana algoritmom pokretnih kocki uz dimenziju rešetke 32, 64 i 128

Kada se algoritam provede za sve točke rešetke, dobiva se mreža trokuta koja opisuje danu izopovršinu. Algoritam je spor, jer je potrebno proći sve ćelije rešetke. Primjerice, za rešetku veličine 64^3 je pri svakom crtanju potrebno provesti algoritam generiranja trokuta oko 250 tisuća puta. Srećom, postupak je moguće ubrzati koristeći GPU što čini primjenu ovog algoritma još interesantnijom za simulacije u stvarnom vremenu. O izvedbi algoritma više u sljedećem poglavlju.

Postoje mnoge varijante koje uvode dodatne slučajeve i tako poboljšavaju ispravnost izlazne mreže trokuta. Neke varijante sažimaju rešetku podataka i značajno ubrzavaju postupak. Jedan odličan primjer je korištenje histopiramida (Dyken, 2008) za kompresiju prostorne rešetke. Histopiramidama se ubrzava dohvat podataka na GPU i omogućuje preskakanje slučajeva u kojima nije potrebno pozivati algoritam kao što su

prazne ili potpuno pune ćelije. Korištenjem ovog postupka uz pokretne kocke moguće je ostvariti brzo crtanje izopovršina jako velikih prostornih rešetki, reda veličine 256^3 upravo zbog činjenice da većina rešetke neće biti popunjena.

4.4 Poligonalna mreža u prostoru pogleda

Ova metoda je relativno nov pristup generiranju poligonalne mreže za neku izopovršinu te je spomenuta u radu kao interesantna alternativa uobičajenim pristupima vizualizacije.

Metoda se temelji na izradi dubinskog polja iz kojeg se algoritmom pokretnih kvadra (2D varijanta pokretnih kocki) stvara 2D poligonalna mreža. Dubine generiranih točaka se podešavaju u prostoru pogleda korištenjem dubinske mape te tako transformiraju natrag u koordinatni prostor svijeta (Müller, 2007.). Iz dobivenih trokuta generiraju se normale za svaki vrh. Na slici 4.7 nalazi se primjer konačne vizualizacije fluida i pripadne generirane poligonalne mreže.

Slika 4.7 Poligonalna mreža u prostoru pogleda [15]

Algoritam se sastoji od 6 koraka:

1. Postavljanje dubinske mape
2. Pronalazak unutarnjih i vanjskih silueta
3. Izglađivanje vrijednosti dubina
4. Izrada poligonalne mreže prilagođenim algoritmom pokretnih kvadra
5. Izglađivanje silueta
6. Pretvorba poligonalne mreže u koordinatni prostor svijeta

Prednost algoritma je što stvaranje dubinske mape i algoritam pokretnih kvadra nisu zahtjevne operacije pa je moguće ostvariti dobre performanse crtanja uz malu cijenu prethodnog računa.

Nedostatak algoritma je što se gube podatci o stražnjim česticama i dubinama što je često bitno u vizualizacijama fluida. Upravo zbog toga, algoritam je primjereniji za vizualizaciju neprozirnih fluida. Naravno, problem je moguće djelomično riješiti izradom poligonalne mreže iz perspektive „iza“ fluida.

Dodatan nedostatak je i činjenica da je izlazna poligonalna mreža valjana samo za određenu poziciju pogleda što može utjecati na postupak bacanja sjena. Rješenje tog problema mogla bi biti provedba algoritma iz pozicije svjetla.

5 Primjer implementacije SPH sustava

Simulacija fluida je računalno jako zahtjevan zadatak i jezik implementacije bi trebao omogućiti što bolje upravljanje podacima, imati brz izvršni kod i imati dobru podršku za povezivanje s grafičkim podsustavom računala. Od popularnih modernih jezika dobar izbor bili bi Java ili C++, a uz današnju kvalitetu njihovih prevodioca razlike u brzinama izvršnog koda trebale bi biti zanemarive. Za potrebe ovog rada implementacija je napravljena u C++ programskom jeziku zbog prethodnog znanja jezika, upoznatosti s grafičkim bibliotekama i pogodnosti jezika matematički zahtjevnim problemima.

Dostupna aplikacijska programska sučelja (API*, engl. Application Programming Interface*) za rad s grafičkim podsustavom su DirectX i OpenGL. DirectX je moguće koristiti samo na Microsoft operativnim sustavima, dok je OpenGL prenosiv i na druge sustave. U implementaciji je korišten OpenGL 4 (uz neke funkcionalnosti ranijih verzija) i pripadajući jezik za sjenčanje GLSL 4 (*engl. OpenGL shading language*).

Korištene su još neke dodatne biblioteke kako bi se olakšala i ubrzala implementacija:

- GLEW, biblioteka koja povezuje i omogućuje pristup funkcijama novijih verzija OpenGL-a koje se temelje na različitim ekstenzijama [24].
- SFML 2.0, biblioteka za upravljanje s prozorima, ulaznim jedinicama, zvukom i mrežom [25]. Prednost ove biblioteke nad sličnima je prenosivost, dobar objektni model i dostupna dokumentacija.
- DevIl 1.7.8, biblioteka koja olakšava učitavanje slika i tekstura u program te korištenje istih u sklopu OpenGL okruženja [26].

Kao razvojno okruženje korišten je Visual Studio 2010 besplatno dostupan studentima kroz MSDNAA program. Korišten je i dodatak Nshader [27] koji omogućuje bojanje izvornog koda programa za sjenčanje i time olakšava njihovu izradu.

5.1 SPH algoritam i potrebni podatci

Kao što je rečeno u poglavlju 2.4 osnovni podatci pohranjeni u česticama su masa, tlak, gustoća, položaj i brzina, a uz njih je potrebno pamtiti i međurezultate simulacije koji se koriste u narednim koracima. Potrebno je pohraniti i silu na česticu, akceleraciju u prethodnom koraku, volumen odnosno radijus čestice, vrijednost gradijenta i laplasijana

polja boja te je pogodno pamtiti i listu susjednih čestica. U tablici 5.1 prikazani su svi interesantni podatci, njihove dimenzije/tip i namjena.

Tablica 5.1 Podatci SPH čestice

Ime	Dimenzija	Namjena
Položaj	Vektor	U gotovo svim izračunima
Brzina	Vektor	Za izračun sile viskoznosti i za integraciju sile
Sila	Vektor	Međusuma u algoritmu i za izračun nove akceleracije
Prethodna akceleracija	Vektor	Korištena pri integraciji sile
Masa	Skalar	Za izračun gustoće i akceleracije
Gustoća	Skalar	Za izračun sile tlaka i volumena
Volumen/radijus	Skalar	Međurezultat za izračun sila
Tlak	Skalar	Korišten za silu tlaka
Gradijent polja boja	Vektor	Detekcija površinskih čestica i računanje sila napetosti
Laplasijan polja boja	Skalar	
Lista susjedstva	Lista	Za ubrzanje postupka

Osnovni algoritam ne razlikuje se mnogo od opisanog SPH postupka. U svojoj osnovici algoritam glasi za svaki korak iteracije:

1. Postavljanje parametara svih čestica na početne vrijednosti
2. Osvježavanje gustoće za svaku česticu:
   - Doprinos drugih čestica
   - Utjecaj čvrstih površina
   - Računanje tlaka svake čestice
3. Računanje sila za svaku česticu:
   - Utjecaj drugih čestica - sile gustoće i viskoznosti
   - Računanje parametara polja boja
4. Određivanje vremenskog koraka
5. Integracija za svaku česticu:
   - Dodavanje sila čvrstih površina i sile napetosti
   - Postupak integracije brzine i pomaka
   - Zadržavanje čestica unutar prostora simulacije

Algoritam 5.1 Osnovni SPH algoritam

Ono što čini implementaciju posebnom su izvedbe pojedinih dijelova. U svakom koraku iteracije moguće je pametno organizirati kod kako bi se smanjio broj ulazaka u petlje, broj usporedbi i račun. O nekim manjim optimizacijama više riječi u sljedećem poglavlju. Slijedi opis pojedinih dijelova algoritma i kako se mogu bolje iskoristiti.

Postavljanje čestica na početne vrijednosti

Ovaj dio algoritma nije kompliciran, ali je nužan. Treba naprosto proći kroz sve čestice i postaviti njihove vrijednosti gustoće, tlaka i laplasijana polja boja, vektore sile i gradijenta polja boja na nulu te isprazniti listu susjedstva. Kako bi se uštedjelo na jednoj petlji, ovaj dio je moguće odraditi u prethodnoj iteraciji, nakon popravljanja čestica koje su izašle iz prostora simulacije. Da bi taj pristup bio valjan, početni parametri novih čestica moraju biti postavljeni na nule kako ne bi došlo do numeričkih problema u prvoj iteraciji u kojoj je nova čestica prisutna u sustavu.

Osvježavanje gustoće

Ovo je zahtjevna petlja koja uspoređuje svaku česticu sa svim ostalim česticama, a ako su dvije čestice susjedne (unutar udaljenosti izglađivanja), međusobno pridonose svojim gustoćama kao što je to opisano u poglavlju 3.2. Pošto su doprinosi gustoća isti za par čestica, potrebno je računati doprinos gustoće samo jednom i dodati ga objema česticama.

Ako se ne koristi neki bolji pristup u ovom je koraku moguće napraviti listu susjedstva za svaku česticu, što će olakšati daljnji rad algoritma jer neće više biti potrebno raditi iteracije kroz sve čestice složenosti$O(n^2)$, već samo $O(mn)$ gdje je m prosječan broj susjednih čestica. Više o pametnijem načinu izrade liste susjedstva u sljedećem poglavlju.

Svakoj se čestici dodaju i vrijednosti gustoće koje uzrokuju čvrste površine koje se nalaze u fluidu. Uobičajeni pristup je da svaka površina doprinosi gustoći čestice kao da se na najbližoj točki površine nalazi neka druga „rubna“ čestica.

Za svaku se česticu također računa i vrijednost tlaka na temelju razlike gustoće u odnosu na gustoću mirovanja kao što je opisano jednadžbom (3-11).

Računanje sila

Opet se posjećuju svi parovi čestica te se za svaki par računaju simetrične sile gustoće i viskoznosti kao što je to opisano jednadžbama (3-15) te (3-17). S obzirom na to

da se opet prolaze svi parovi čestica usput je poželjno računati i vrijednosti vezane uz polje boja jednadžbama (3-18) i (3-19).

Određivanje vremenskog koraka

Vremenski korak moguće je odrediti i izvan algoritma, ali ako se koristi Courant-Friedrichs-Lewy uvjet stabilnosti (Desbrun, 1996.) onda je, zbog prisutnosti parametra brzine čestice, to lakše ostvariti unutar petlje algoritma.

Druga mogućnost je koristiti kao vremenski korak vrijeme jednog ciklusa iteracije SPH algoritma i vizualizacije SPH čestica, odnosno trajanje prolaza kroz petlju čitavog programa. S obzirom na to da će vrijeme simulacije biti usklađeno sa stvarnim protokom vremena ovaj će pristup dati najvjernije rezultate. Nedostatak ješto za malo veći broj čestica, jedna iteracija kroz SPH algoritam troši puno vremena pa bi vremenski korak postajao prevelik kako broj čestica i složenost sustava raste. Uzevši tu činjenicu u obzir, ovaj se pristup nikada ne susreće u praksi.

Treća mogućnost je koristiti konstantan vremenski korak. Problem s tom opcijom je odabir dobrog koraka, a dobar red veličine je do 10 milisekundi. Ovaj je pristup čest jer ne zahtjeva puno dodatnog računa, a daje dobre rezultate.

Integracija sila

Ponovno se za svaku česticu posjećuju čvrste površine, samo što se ovaj puta računaju sile kojima površine djeluju na čestice koristeći jednadžbe (3-23) i (3-24). Ako su površine pokretne, u ovom se koraku računaju i sile na površine te se integrira i njihov pomak.

Ako je zadovoljen uvjet praga polja boja da je gradijent dulji od neke granične vrijednosti (slika 3.1.) tada se sili na česticu pridodaje i sila napetosti koja je u suprotnom smjeru od smjera gradijenta polja boja, a dana je izrazom (3-21).

Konačno se korištenjem neke od metoda integracije iz sile izražava akceleracija te se računa nova brzina i nova pozicija čestice. Po potrebi, moguće je i prigušiti brzinu svake čestice kako bi se prividno povećala stabilnost simulacije.

Čestice koje su izašle izvan prostora simulacije potrebno je vratiti. To je moguće tako da se jedna od čvrstih površina modelira kao kocka poravnata s koordinatnim osima (AABB, *engl. axis aligned bounding box*) koja ne dopušta česticama da izađu izvan nje.

Kada neka od čestica prođe granicu površine, odmakne se za udaljenost prodora te se njen vektor brzine zrcali od površine kao što je opisano jednadžbom (3-25).

Prikaz čestica

Na kraju algoritma potrebno je koristeći nove pozicije čestica prikazati fluid nekom od metoda vizualizacije. Izlaz algoritma su pozicije čestica te radijusi čestica za koje se pretpostavlja da su sfere. Kao radijus moguće je napraviti neku transformaciju udaljenosti izglađivanja te će tada sve čestice imati istu veličinu. Bolji bi pristup bio koristiti volumen svake čestice za dobivanje radijusa:

$$V = \frac{4}{3}\pi r^3$$

Problem je u tome što se radijus dobiva iz izraza za volumen traženjem trećeg korijena, što je računalno skupa operacija. Ali za manji broj čestica utjecaj je zanemariv.

5.2 Optimizacije algoritma

Neke od mogućih optimizacija su podjela prostora u rešetku kako bi se lakše pronalazile susjedne čestice, optimizacija izvornog koda kako bi se dobio efikasniji izvršni kod, izbacivanje i izmjena matematičkih operacija unutar algoritma te paralelizacija.

Prostorna rešetka susjedstva

Najefikasnija metoda optimizacije SPH metode je optimizacija pretrage susjedstva i stvaranje liste susjedstva. Kao što je ranije opisano, listu susjedstva je moguće oformiti prilikom izračuna gustoće kada se prvi puta u iteraciji algoritma vrši račun nad parovima čestica. U primjeru implementacije uz ovaj rad korišten je upravo ovaj pristup. Svaka čestica sadrži listu pokazivača na njoj susjedne čestice. Lista se gradi prilikom prvog obilaska svih čestica u svakoj iteraciji algoritma.

Bolji pristup bio bi na neki način implicitno dobiti znanje o susjednim česticama kako bi izgradnja listi susjedstva mogla prethoditi koraku računanja gustoće. To je moguće korištenjem prostorne rešetke čije su ćelije dimenzija ne manjih od duljine izglađivanja (Müller, 2003.). Svaku je česticu potrebno usporediti s česticama u susjednim ćelijama i s česticama u vlastitoj ćeliji, što znači obilazak 27 ćelija u najgorem slučaju. S obzirom na to da se sile između čestica računaju simetrično, u stvari je potrebno posjetiti samo pola susjedstva i vlastitu ćeliju odnosno 14 ćelija. Veći problem ovog pristupa je efikasno indeksiranje susjednih ćelija.

Ručno računanje koordinata i izgradnja listi može biti složen postupak te je s ciljem jednostavnijeg pronalaska susjedstva Teschner (2003.) opisao metodu optimiziranog sažetka prostornih koordinata. Tom se metodom svaka čestica obrađuje funkcijom sažimanja (*engl. hash function*) i tako se dobiva indeks odnosno oznaka ćelije unutar mape sažetaka (*engl. hash map*) kojoj neka čestica pripada. Dodatna prednost je što se ćelije prostora u kojima nema čestica neće popunjavati, tj. ne zauzimaju prostor u memoriji.

Bolji pristup opisan je u (Onderik, 2007.), a razvijena je metoda indeksiranja ćelija (*engl. cell indexing*) koja pokazuje bolje rezultate brzine nad metodom sažetaka. Princip je sličan, za svaku se česticu računa njen ključ koji je za razliku od sažetka jedinstven svakoj čestici. Set ključeva se zatim uređuje korištenjem *radix* algoritma uz $O(n)$ složenost. To je moguće postići ovim algoritmom kada je poznata duljina ključeva kao što je slučaj u ovom algoritmu. U poredanom nizu indeksa jednostavno je pronaći susjedne ćelije i time susjedne čestice.

Optimizacija izvornog koda

Programiranje u jeziku C++ nije jednostavno i postoji mnogo načina na koje je moguće napisati kod koji radi isti posao, ali se prevodi u brži izvršni kod. Neke su metode pogodnije za algoritme kao što je SPH pa ih je interesantno napomenuti jer mogu donijeti značajno ubrzanje.

Za velike količine podataka koje se obrađuju, bolja brzina obrade postiže se kada su strukture u kojima su podatci pohranjeni veličinom u oktetima poravnate na neku od potencija broja 2. Tako SPH algoritam koji zaokružuje veličinu strukture čestice može postići ubrzanje u odnosu na strukturu koja nema lažni dodatak memorije. U primjeru implementacije veličina temeljne strukture je 96 okteta te se s njom postiže prosječna brzina računanja jedne iteracije od 0.0216 sekundi za 1000 čestica. Kada se koristi struktura poravnata na 128 okteta za isti broj čestica postiže se prosječna brzina od 0.0200 sekundi, što je 7.5% manje od korištenja neporavnate strukture.

Još jedna korisna optimizacija je korištenje *inline* ključne riječi prilikom deklaracije funkcije. Ta ključna riječ je preporuka prevodiocu da izvršni kod deklarirane funkcije ubaci na mjesto njena pozivanja [30]. Ako prevodilac obavi ovaj postupak, štedi se na pozivu funkcije što može biti značajno za funkcije koje se često pozivaju. U SPH algoritmu

jezgrene funkcije odgovaraju modelu funkcije koje su idealne za *inline* postupak: kratke su, jednostavne, nisu rekurzivne i često se pozivaju. Utjecaj ove optimizacije je teško primijetiti jer prevodilac kojiput sam odabire funkcije čije će tijelo ubaciti na mjesto pozivanja bez programerove prethodne upute.

Optimizacija matematike

Neke dijelove računa je moguće skratiti i preoblikovati kako bi se izvršavao što manji broj operacija. Iako ušteda ne djeluje veliko, za veći broj čestica moguće je ostvariti pozitivan utjecaj na performanse. Još jedna prednost optimizacije matematike je smanjivanje grešaka nastalih uoči nepreciznosti računa brojeva s posmačnim zarezom. Ako u programu ima manje operacija, manje su vjerojatnosti grešaka.

Ako pretpostavimo da je masa svih čestica ista onda zbog zakona distributivnosti jednadžbe za izglađena svojstva (3-5), (3-6) i (3-7) možemo zapisati na sljedeći način:

$$A_\mathrm{i} = \mathrm{m}\sum_{\mathrm{j}\neq\mathrm{i}}\frac{A_j}{\rho_j}W\left(|\vec{r}_\mathrm{i}-\vec{r}_\mathrm{j}|,h\right)$$

$$\nabla A_\mathrm{i} = \mathrm{m}\sum_{\mathrm{j}\neq\mathrm{i}}\frac{A_j}{\rho_j}\nabla W\left(|\vec{r}_\mathrm{i}-\vec{r}_\mathrm{j}|,h\right)$$

$$\nabla^2 A_\mathrm{i} = \mathrm{m}\sum_{\mathrm{j}\neq\mathrm{i}}\frac{A_j}{\rho_j}\nabla^2 W\left(|\vec{r}_\mathrm{i}-\vec{r}_\mathrm{j}|,h\right)$$

To zapravo znači da možemo u svim izrazima koji sadrže neku od ovih jednadžbi izlučiti masu kao faktor konačnog izraza, odnosno da ukupan izraz možemo pomnožiti s iznosom mase čestice na kraju sumacije, samo jednom umjesto pri svakom izračunu izglađenog svojstva. To je moguće primijeniti na računanje gustoće gdje je nužno pomnožiti iznos sume s masom čestice. U slučaju računanja sile, faktor mase je moguće u potpunosti izbaciti jer se sila na kraju koristi za izračun akceleracije (3-26) pa je u tom slučaju akceleracija jednaka izračunatom vektoru „sile“ jer više nije potrebno dijeliti s masom.

Postoji još jedna mogućnost pokrate na izrazima za izglađene vrijednosti, koja može zamijeniti operaciju dijeljenja s množenjem. Iako nije jednostavno odrediti na modernim arhitekturama i za sve situacije postoji li značajna razlika u brzini dijeljenja i množenja, u SPH sustavu je riječ o puno operacija dijeljenja i množenja pa optimizacija koja može zamijeniti povijesno sporiju operaciju s bržom nije tako loša ideja. Riječ je o izrazu $\frac{m_j}{\rho_j}$ koji se nalazi u svim jednadžbama za izglađivanje. Naime taj izraz u stvari opisuje

volumen pojedine čestice izražen preko jednadžbe gustoće. Volumen čestice je moguće izračunati jednom, nakon što je izračunata gustoća čestice. Tada u izrazima (3-5), (3-6) i (3-7) možemo uvesti pokratu:

$$A_\mathrm{i} = \sum_{\mathrm{j}\neq\mathrm{i}} \mathrm{V_j}\, A_j W\left(|\vec{r}_\mathrm{i} - \vec{r}_\mathrm{j}|, h\right)$$

$$\nabla A_\mathrm{i} = \sum_{\mathrm{j}\neq\mathrm{i}} \mathrm{V_j}\, A_j \nabla W\left(|\vec{r}_\mathrm{i} - \vec{r}_\mathrm{j}|, h\right)$$

$$\nabla^2 A_\mathrm{i} = \sum_{\mathrm{j}\neq\mathrm{i}} \mathrm{V_j}\, A_j \nabla^2 W\left(|\vec{r}_\mathrm{i} - \vec{r}_\mathrm{j}|, h\right)$$

Izrazi su i dalje istovjetni onima prije, samo nije potrebno dijeliti s nekim brojem, već množiti. Ako i u ovom slučaju pretpostavimo da su mase jednake, možemo izbjeći dijeljenje s masom pri izračunu akceleracije tako da volumen računamo samo kao $\frac{1}{\rho_j}$.

Osim računskih pokrata, poželjno je koristiti i izraze za simetrične sile kako bi se za jedan par čestica vrijednost sile računala samo jednom.

Slijedi konačan pregled svih pokrata i smanjenog algoritma za izračun sila.

Simetrični izrazi za sile (3-15) i (3-17) glase, uz uvođenje volumena kao parametra:

$$\vec{f}_i^{\,tlak} = -\sum_{j\neq i} V_j \frac{p_i + p_j}{2} \nabla W\left(|\vec{r}_i - \vec{r}_j|, h\right)$$

$$\vec{f}_\mathrm{i}^{\,\mathrm{viskoznost}} = \mu \sum_{\mathrm{j}\neq\mathrm{i}} V_j (\mathrm{v}_j - \mathrm{v}_\mathrm{i})\, \nabla^2 W\left(|\vec{r}_\mathrm{i} - \vec{r}_\mathrm{j}|, h\right)$$

Za svaki par čestica (i, j) gdje $i \neq j$ računamo sile kao:

$$\vec{f}_i = \vec{f}_i^{\,t} + \vec{f}_\mathrm{i}^{\,\mathrm{v}} = V_j \left(-\tfrac{p_i+p_j}{2} \nabla W\left(\left|\vec{r}_i - \vec{r}_j\right|, h\right) + \mu(\mathrm{v}_j - \mathrm{v}_\mathrm{i}) \nabla^2 W\left(\left|\vec{r}_\mathrm{i} - \vec{r}_\mathrm{j}\right|, h\right)\right) \quad (5\text{-}1)$$

$$\vec{f}_j = \vec{f}_j^{\,t} + \vec{f}_\mathrm{j}^{\,\mathrm{v}} = V_i \left(-\tfrac{p_i+p_j}{2} \nabla W\left(\left|\vec{r}_j - \vec{r}_i\right|, h\right) + \mu(\mathrm{v}_i - \mathrm{v}_\mathrm{j}) \nabla^2 W\left(\left|\vec{r}_j - \vec{r}_i\right|, h\right)\right) \quad (5\text{-}2)$$

Zato što je ulaz u jezgrene funkcije apsolutna vrijednost (odnosno odabrane su parne jezgre, poglavlje 3.4) moguće je iz (5-1) i (5-2) izlučiti zajedničke, simetrične izrazi za tlak i viskoznost. Potrebno je obratiti pozornost kako su izrazi za viskoznost u (5.1) i (5.2) suprotnog predznaka jer se vektori brzina oduzimaju drugim redoslijedom:

$$\vec{T} = -\frac{p_i + p_j}{2} \nabla W\left(\left|\vec{r}_i - \vec{r}_j\right|, h\right)$$

$$\vec{V} = \mu(\mathrm{v}_j - \mathrm{v}_\mathrm{i}) \nabla^2 W\left(\left|\vec{r}_\mathrm{i} - \vec{r}_\mathrm{j}\right|, h\right)$$

Uvrštavanjem natrag u (5-1) i (5-2) uz oprez pri predznacima uz izraz viskoznosti:

$$\vec{f}_i = \vec{f}_i^t + \vec{f}_\mathrm{i}^\mathrm{v} = V_j(\vec{\mathrm{T}} + \vec{\mathrm{V}}) \tag{5-3}$$

$$\vec{f}_j = \vec{f}_j^t + \vec{f}_\mathrm{j}^\mathrm{v} = V_i(\vec{\mathrm{T}} - \vec{\mathrm{V}}) \tag{5-4}$$

Sličnim se postupkom dobiju i zajednički izrazi za gradijent i laplasijan polja boja za par čestica iz jednadžbi (3-18) i (3-19):

$$C_{gradijent} = W(|\vec{r}_\mathrm{i} - \vec{r}_\mathrm{j}|, h)$$

$$C_{laplasijan} = \nabla W(|\vec{r}_\mathrm{i} - \vec{r}_\mathrm{j}|, h)$$

$$\mathrm{c_i} = \mathrm{V_j} C_{gradijent}$$

$$\nabla c_\mathrm{i} = \mathrm{V_j} C_{laplasijan}$$

$$\mathrm{c_j} = \mathrm{V_i} C_{gradijent}$$

$$\nabla c_\mathrm{j} = \mathrm{V_i} C_{laplasijan}$$

Dobiveni se izrazi koriste za izračun sile napetosti kada je to potrebno.

U konačnici se računa izraz za akceleraciju (3-26) u kojem se naprosto komponente vektora sile podijele s iznosom mase čestice. Prethodnim izrazima je pokazano kako je dio svake sile množenje s volumenom čestice. Za računanje volumena koristi se masa čestice ista za sve čestice i gustoća specifična za svaku česticu. To znači da će se faktor mase skratiti prilikom računanja akceleracije, što znači da volumen možemo izračunati kao:

$$V_i = \frac{1}{\rho_i}$$

A akceleracija je tada:

$$\vec{a}_i = \vec{f}_i$$

Iako utjecaj na brzinu izvođenja neće nužno biti značajan, napravljenim transformacijama izvorni kod programa je kraći i pregledniji, a smanjen je broj operacija što nikako ne može biti loše u matematički zahtjevnom postupku kao što je SPH algoritam.

Paralelizacija algoritma

Problem paralelizacije SPH algoritma je tema mnogih istraživanja i radova. Moguće je napraviti inačicu algoritma prilagođenu za paralelizaciju na CPU, ali i za GPU. U oba slučaja potrebno je obratiti najviše pozornosti na pristup podatcima.

Sva paralelna rješenja zahtijevaju podjelu prostora simulacije u rešetku te dodjeljivanje čestica ćelijama. Račun unutar svake ćelije je tada odvojen od susjednih ćelija te ga je moguće ostvariti paralelno s minimalnim preklapanjem. Potrebno je obratiti pozornost na računanje simetričnih vrijednosti kod kojih može doći do kolizija u pisanju podataka ako je memorija dijeljena. Riječ je o slučaju kada dvije čestice pišu u memoriju zajedničke susjedne čestice. Najgori ishod tog događaja je da jedan od pribrojenih iznosa neće biti upisan i da će čestica imati malo netočno ponašanje. Jedno od rješenja tog problema je korištenje kopija memorije, što je slučaj u GPU implementacijama gdje do ovakvih kolizija ne dolazi.

Usko grlo paralelizacije SPH algoritma je izrada liste susjedstva koja se izrađuje nekom od metoda pretrage prostorne rešetke spomenutom ranije u ovom poglavlju. U slučaju paralelizacije na GPU liste indeksa susjednih čestica pohranjuju se u teksturu te su kao takve jednostavno dostupne unutar programa za sjenčanje.

Detaljnije o paralelizaciji SPH algoritma govore Goswami (2010.) i Harada (2008.), a moguće je pronaći i puno drugih primjera izvedbe kako na CPU, tako i na GPU.

5.3 SPH parametri

Jedan od problema prilikom implementacije SPH sustava je odabir parametara simulacije kako bi se ostvario fluid željenih karakteristika. Ponekad neka kombinacija parametara nije pogodna za sustav i moguća su nepredviđena ponašanja. Razlog tome su različitosti u implementacijama, redoslijedu operacija, dimenzijama prostora i dosljednosti algoritmu što najčešće znači da parametri jedne simulacije, neće rezultirati istim ponašanjem fluida u drugoj simulaciji.

Jedan od razloga zašto su parametri nedosljedni je podjela prostora odnosno dimenzionalnost prostora. Primjerice, SPH simulacija u tri dimenzije nalazi se unutar prostora omeđenog kockom čiji se krajnji vrhovi nalaze u točkama $(0,0,0)$ i $(10,10,10)$. Da li to znači da je prostor fluida $1000m^3$ ili da je prostor fluida zapremine jedne litre $1000cm^3$? Razlika je pozamašna, jer u prostoru zapremine jedne litre, fluid gustoće $1\frac{kg}{l}$ i masom jedne čestice $0.01kg$ može sadržavati svega 100 čestica, dok u slučaju mjerne jedinice metra, fluid može sadržavati 10^8 čestica. To znači da će na velikom prostoru biti potrebno smanjiti gustoću ili povećati masu, a na manjem prostoru treba povećati gustoću ili smanjiti masu čestice kako bi se ostvario željeni efekt.

U ovom jednostavnom primjeru nije bilo riječi o ostalim parametrims čiji utjecaj u izračunu ovisi o udaljenostima i o dimenzijama. Primjerice prevelika plinska konstanta može uzrokovati prevelike tlakove i samim time iznimno kaotično ponašanje fluida. Velika viskoznost na velikom prostoru može uzrokovati potpuni zastoj i grupiranje čestica u male, nepomične grupe.

Duljina izglađivanja je jedan od najbitnijih parametara SPH algoritma jer određuje na kojoj će se udaljenosti promatrati susjedstvo neke čestice. Ako je duljina prevelika, provoditi će se previše računanja za udaljene čestice, čiji utjecaj nije dovoljno velik da bi bio bitan. S druge strane, premalena duljina izglađivanja će stvoriti privid ponašanja fluida kao da su čestice nezavisne.

Parametrima sile površinske napetosti moguće je regulirati kako će se kruto ponašati fluid. Ako je utjecaj sile napetosti prejak, fluid će se grupirati u nakupine koje će se odupirati raspadanju na manje dijelove pod utjecajem dolaska novih čestica. Efekt će biti sličan onome velike gustoće. Uz preslab utjecaj sile napetosti moguće je „bježanje“ čestica.

U tablici 5.2 navedeni su parametri korišteni u ovoj implementaciji, kao i donja te gornja granica pojedinog parametra. Prostor je ograničen AABB kockom između točaka$(0,0,0)$ i $(10,10,10)$.

Tablica 5.2 Parametri simulacije

	Početno	**Najmanje**	**Najviše**
Plinska konstanta k	**0.8314**	0.05	10
Duljina izglađivanja h	**1.5**	0.1	10
Gustoća ρ_0	**0.06**	0.001	3
Masa čestice m	**0.05**	0.001	3
Dinamička viskoznost μ	**0.0894**	0.001	3
Koeficijent površinske napetosti σ	**0.015**	0.005	1
Prag polja boja c_{prag}	**0.27**	0.005	1

Još jedan jako bitan parametar simulacije je i vremenski korak, a mogućnosti odabira opisane su u poglavlju 3.7. U primjeru implementacije korišten je konstantni korak od 0.001 sekunde.

5.4 Interakcija

Interakcija je vanjski utjecaj na SPH sustav koji može biti uzrokovan prisutnošću oblika, fizikalnih utjecaja unutar fluida te dodavanja ili oduzimanja čestica.

Površine

Površine interakcije se koriste kako bi se ogradio prostor simulacije i kako bi se ostvarila interakcija fluida s drugim modelima u sceni. Problem korištenja površina je što su njihove dimenzije varijabilne i nije ih moguće smjestiti u prostornu rešetku kao što je to moguće s česticama. To znači da je svaku česticu potrebno testirati sa svakom površinom kako bi se utvrdila interakcija. Kao što je opisano u poglavlju 3.5 površine djeluju izglađenim silama na čestice jer se modeliraju kao da su sačinjene od čestica. Razlika je u tome što površine ne dopuštaju čestici da prođe kroz nju tako što vrše jednostavnu refleksiju vektora brzine.

Interesantne su površine koje je jednostavno matematički opisati i koje se mogu služiti kao poopćenje za složenije objekte. Najjednostavnije i najinteresantnije površine su: ravnina, pravokutnik poravnat s koordinatnim osima, općeniti pravokutnik, sfera, cilindar poravnat s osima, općeniti cilindar, otvorena sfera, otvoreni cilindar. Pregled površina i podataka koji se koriste dan je u tabeli 5.3. Osim navedenih podataka, koristan je podatak i debljina površine.

Tablica 5.3 Razne površine i podatci koji ih opisuju

Površina	**Podatci**
Ravnina	Jedna točka ravnine i normala ravnine
Pravokutnik poravnat s koordinatnim osima	Dva dijagonalna vrha pravokutnika
Općeniti pravokutnik	Svi vrhovi pravokutnika ili ravnine koje opisuju sve plohe pravokutnika
Sfera	Centar i radijus
Cilindar poravnat s koordinatnim osima	Visina donje plohe, visina gornje plohe i radijus
Općeniti cilindar	Centar donje plohe, centar gornje plohe i radijus
Otvorena sfera	Centar, radijus, vektor otvora i visina na kojoj je otvor
Otvoreni cilindar	Isti parametri kao i za prethodne cilindre, samo se zanemaruje gornja ploha

Interesantan dodatak implementaciji su i povratne sile na površine kojima se može ostvariti i utjecaj fluida na predmet koji se nalazi u fluidu. U tom slučaju potrebno je dobro obratiti pozornost na to kako će se definirati parametri površine, poput mase i gustoće te kako će površina vršiti interakciju s fluidom.

Zone sila

Zone interakcije su područja sila kojima bi bilo moguće simulirati razne vanjske utjecaje poput udara, vjetra i topline. Zona bi bila definirana na isti način kao i površina s razlikom što čestica koja prolazi kroz zonu ne bi sudjelovala u sudaru, već bi na nju djelovala neka sila.

Primjerice kako bi se simulirao tok fluida bez dodavanja novih čestica, na jedan kraj spremnika fluida mogla bi se staviti zona utjecaja horizontalne sile. Za simulaciju ključanja tekućine ili sličnog fenomena, zona sile bila bi postavljena na dno spremnika.

Izvori i ponori čestica

Ova su područja mjesta gdje čestice nastaju i ubacuju se u sustav ili nestaju i brišu se iz sustava. Korištenjem ovog pristupa moguće je ostvariti scene poput toka tekućine iz slavine i otjecanje u odvod.

Površine izvora i ponora mogu biti 2D plohe ili ravnine, a mogu biti definirane i kao 3D objekti unutar koji nastaju/nestaju čestice. Korisne mogućnosti izvora su definiranje nasumičnog stvaranja, raspršivanja i varijabilne brzine izlaza čestica.

5.5 Vizualizacija

U sklopu implementacije isprobane su dvije metode vizualizacije: polje sličica i pokretne kocke. S ciljem ubrzanja napravljene su varijante algoritma za GPU koje koriste programe za sjenčanje, odnosno sjenčanje geometrije dostupno s OpenGL 3.2 API-jem.

Polje sličica

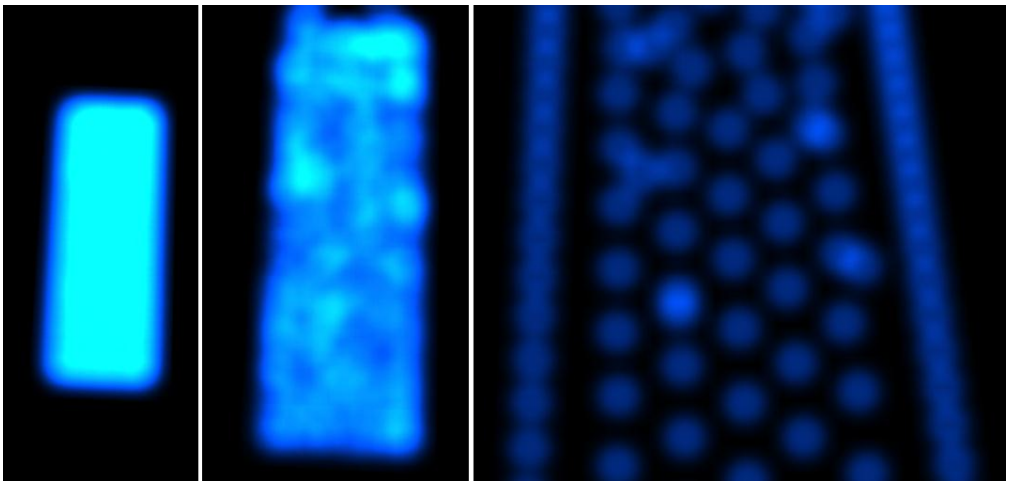
Polje sličica moguće je isprogramirati tako da se „ručno“ stvaraju kvadri, orijentiraju prema gledatelju i teksturiraju. Taj pristup je jako spor i ne pruža mnogo mogućnosti i slobode prikaza, te stoga nije razmatran.

Kao što je spomenuto u poglavlju 4.2 postoji ugrađena funkcionalnost u OpenGL standardu koja omogućuje jednostavan prikaz polja sličica korištenjem funkcionalnosti cjevovoda. Potrebno je definirati parametre točaka, veličinu točke i okruženje teksture. Nakon omogućavanja polja sličica potrebno je samo poslati podatke o pozicijama čestica. Ova metoda ne dopušta varijabilne veličine sličica, ali je zato teksturu moguće kombiniratis bojom i ostvariti skaliranje sličice s obzirom na udaljenost od kamere. U polju algoritam 5.2 nalazi se isječak koda kojim se koristi funkcionalnost OpenGL-a.

Postavljanjem parametra `GL_POINT_DISTANCE_ATTENUATION` određuje se promjena veličine sličice obzirom na udaljenost od kamere prema formuli:

$$faktor = \frac{1}{a + b * d + c * d^2}$$

gdje su a, b i c parametri, a d je udaljenost. Na slici 5.1 nalazi se prikaz fluida poljem sličica čije se veličine ne skaliraju dovoljno dobro za veliku blizinu prikaza. Lijeva slika je iz najveće udaljenosti, dok je desna iz najmanje udaljenosti. Iako prikaz na malim udaljenostima nije zadovoljavajuć, koristan je za promatranje ponašanja i rasporeda čestica u fluidu. Na slici 5.1 je tako vidljiv pravilan, heksagonalan raspored čestica, a moguće je bolje promotriti i druge fenomene kao što su valovi ili nepravilnosti u radu SPH sustava.



Slika 5.1 Razlika u prikazu zbog neskaliranih veličina čestica

Pozivanje `glPointSize` je nužno kako bi se postavila veličina sličice, a ako taj parametar nije definiran, izlaz crtanja biti će kvadrat veličine jednog piksela na mjestu sličice. Ostali postavljeni parametri reguliraju najveću i najmanju veličinu sličice te prag zagušenja. Crtanje polja sličica obavlja se pozivanjem `glEnable` s parametrom `GL_POINT_SPRITE` te crtanjem polja točaka u kojemu su pozicije čestica.

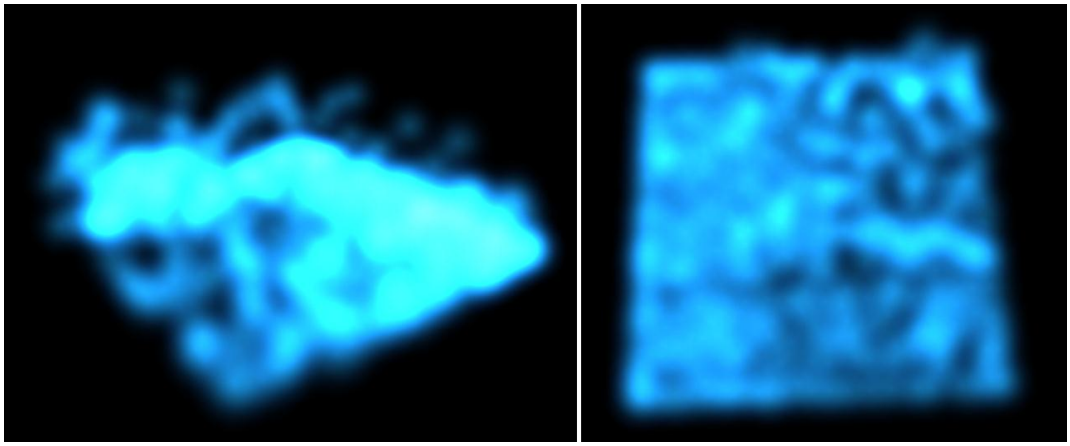
```
floatattenuation[] =  { 0.4, 0.01f, 0.0f };
glPointParameterfv( GL_POINT_DISTANCE_ATTENUATION, attenuation);

floatmaxSize = 0.0f;
glGetFloatv( GL_POINT_SIZE_MAX, &maxSize );
glPointSize( size*10 >maxSize ? maxSize : size*10 );

glPointParameterf( GL_POINT_FADE_THRESHOLD_SIZE, 60.0f );
glPointParameterf( GL_POINT_SIZE_MIN, 1.0f );
glPointParameterf( GL_POINT_SIZE_MAX, maxSize );

glColor3f( color.r, color.g, color.b );
glTexEnvf( GL_POINT_SPRITE, GL_COORD_REPLACE, GL_TRUE );

glEnable( GL_POINT_SPRITE );
// drawpoints
glDisable( GL_POINT_SPRITE );
```

Algoritam 5.2 Isječak koda koji koristi ugrađenu funkcionalnost polja sličica

Za razliku od korištenja funkcionalnosti cjevovoda, prikaz polja sličica korištenjem programa za sjenčanje dopušta veću fleksibilnost za postizanje posebnih efekata. Princip programa za sjenčanje je jednostavan:

1. Sjenčanje vrhova: Transformacija u prostor pogleda.
2. Sjenčanje geometrije: Stvaranje 4 nova vrha koji čine kvadar oko ulaznog vrha, generiranje koordinata teksture te perspektivna projekcija.
3. Sjenčanje fragmenata: Čitanje teksture, izvođenje željenog efekta.

Osim matrica transformacija i teksture, potreban je samo parametar veličine. Njega je kao i u ugrađenoj funkcionalnosti moguće mijenjati s obzirom na udaljenost. Na slici 5.2 nalazi se prilaz fluida poljem sličica koje stvara i crta program za sjenčanje.

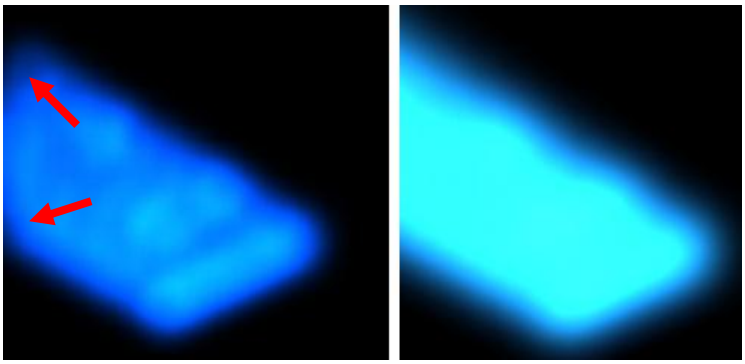


Slika 5.2 Polje sličica nacrtano korištenjem progama za sjenčanje geometrije

U usporedbi sa slikom 4.4 gdje se nalazi prikaz polja sličica korištenjem ugrađene funkcionalnosti, bitnijih razlika nema. Naravno, razlike postoje na različitim udaljenostima

kao što je prikazano na slici 5.1, ali to je moguće nadoknaditi dobrim odabirom parametara.

Razlika u brzini između ugrađene funkcionalnosti i sjenčanja u najboljem slučaju nema, a u većini situacija je ugrađena funkcionalnost brža i to posebice za manji broj čestica. Iako postoji mala prednost u brzini, ugrađeni prikaz ima veliki nedostatak s odbacivanjem čestica na rubu pogleda. Naime pri odbacivanju se promatra samo položaj čestica, a ne i njihova veličina. Taj problem nije prisutan kada se koristi sjenčanje jer je potrebno ručno regulirati koja će geometrija biti odbačena. Slika 5.3 prikazuje razlike u prikazu na rubu ekrana za ugrađeni prikaz i prikaz programom za sjenčanje. Crvenom strelicom je označeno gdje su nepotrebno odbačene čestice.



Slika 5.3 Primjer greške odbacivanja elemenata polja sličica na rubu prikaza

Zanimljivo je napomenuti kako je prikaz polja sličica fiksnim cjevovodom izbačen iz definicije OpenGL API-ja u verziji 3.3 te korištenje tih funkcionalnosti nije preporučeno, iako je moguće.

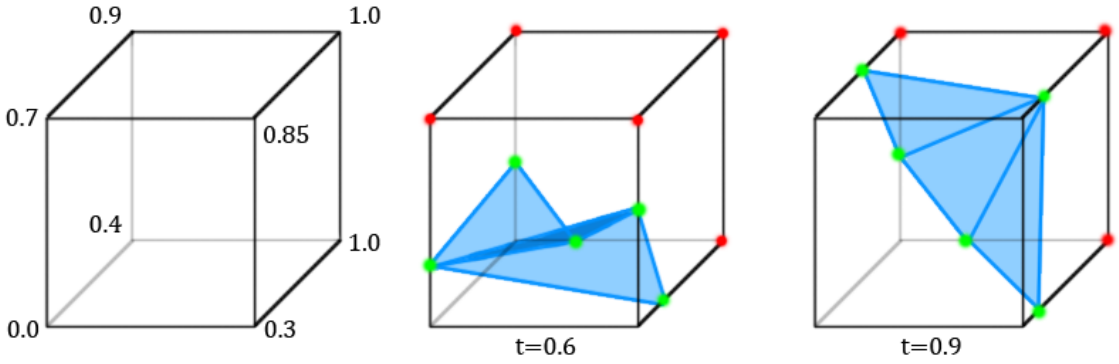
Pokretne kocke

Metoda pokretnih kocki je u svojoj osnovnoj verziji spor algoritam koji stvara poligonalnu mrežu koja nije primjerena za prikaz oblih površina kao što je to fluid. Nažalost, rezultat prikaza sfere pokretnim kockama kao što to prikazuje slika 4.6 nije zadovoljavajući za prikaz fluida zbog premale zaobljenosti izlazne mreže.

Za potrebe generiranja prikladnije poligonalne mreže, vrhovi stvorenih trokuta interpoliraju se iz vrhova ćelije podatkovne rešetke, bez upotrebe tablice bridova.

U prostornoj rešetci podataka promatraju se ćelije omeđene s osam okolnih točaka. Za svaku ćeliju određuje se na temelju referentne vrijednosti praga ulazi li jedna

od pripadnih točaka ćelije u interpolaciju. Na slici 5.4 nalazi se primjer odabira točaka na temelju različitih vrijednosti rešetke i na temelju različitih vrijednosti praga.



Slika 5.4 Utjecaj vrijednosti praga na odabir točaka

Kada su određene točke koje ulaze u algoritam (označene crveno), pronalaze se interpolirane točke na bridovima kocke (označene zeleno). Te točke će biti vrhovi trokuta u izlaznoj poligonalnoj mreži (označeni plavo). Interpolacija je linearna, a koeficijent se računa na temelju udaljenosti od referentne vrijednosti:

$$V_{izlaz} = V_1(1-a) + V_2 a$$

$$a = \frac{P - v_1}{v_2 - v_1}$$

gdje su V_1 i V_2 točke u 3D prostoru rešetke koje predstavljaju vrhove kocke, v_1 i v_2 su njihove pripadne vrijednosti, a P je vrijednost praga. Na slici 5.4 zelene točke se ne nalaze na sredini brida već su pomaknute ovisno o vrijednostima vrhova. Korištenjem ove metode nije potrebno računati i koristiti tablicu bridova, a izlazni ćetrokuti činiti površinu koja bolje prati konturu izopovršine. Nakon što su generirane interpolirane točke bridova, korištenjem tablice trokuta stvaraju se veze između tih točaka, a time i izlazni trokuti (poglavlje 4.3).

Sama prostorna rešetka je niz brojeva s posmačnim zarezom koji predstavljaju skalarno polje prisustva ili blizine čestice fluida. Postupak je opisan u poglavlju 4.3. Popunjavanje prostorne rešetke moguće je za svaku točku rešetke ili za svaku česticu fluida. U implementaciji je korišten pristup superponiranja sfere čestice fluida na rešetku. Kako bi se dobilo izglađenije polje i izbjegli vizualni artefakti, poželjno je pribrajati vrijednosti utjecaja i ograničiti rezultat nekom maksimalnom vrijednošću te poljem praga regulirati konačan izgled fluida.

Ubrzanje postupka moguće je particioniranjem prostora na dijelove u kojima je potrebno stvarati trokute i na dijelove u kojima to nije potrebno činiti. Rješenje je korištenje strukture nalik oktalnom stablu kojom se pamte potprostori u kojima je potrebno provoditi algoritam. Ovaj postupak je isplativ jer fluid ima tendenciju grupiranja, što znači da će potencijalno velika područja biti prazna i tamo algoritam nije potreban.

Bolji način ubrzanja je korištenje programa za sjenčanje, odnosno sjenčanja geometrije za generiranje trokuta. Osim što će postupak biti brži, moguće je jednostavnije ostvariti i interesantne efekte poput osvjetljenja, artefakta refleksije i loma svjetlosti, preslikavanja okoline, prozirnosti i slično.

Transformacija algoritma na GPU zahtjeva nekoliko prilagodbi podataka s kojima algoritam radi:

- Pretvorba tablice trokuta u teksturu kako bi joj se moglo pristupiti unutar programa za sjenčanje
- Pretvorba prostorne rešetke u 3D teksturu
- Stvaranje polja točaka koje će biti ulaz u program za sjenčanje. To polje točaka predstavlja pozicije ćelija unutar prostorne rešetke skalarnog polja.

Kako bi se podatci pravilno uzorkovali potrebno je ispravno podesiti parametre teksture. Oni određuju kako se interpretiraju podatci u teksturi i kako se vrši njihova interpolacija.

U polju algoritam 5.3 nalaze se postavke za teksturu tablice trokuta. Tekstura je podešena tako da se uzimaju najbliže vrijednosti teksture (bez interpolacije) te da se izvan područja teksture uzimaju najbliže rubne vrijednosti (`GL_CLAMP_TO_EDGE`). Taj pristup je nužan jer su potrebne točne vrijednosti iz tablice. Vrijednosti tablice trokuta se pohranjuju kao cjelobrojne vrijednosti u *alpha* komponenti boje.

```
glBindTexture(GL_TEXTURE_2D,triangeTableTexture);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_R, GL_CLAMP_TO_EDGE);

glTexImage2D( GL_TEXTURE_2D, 0, GL_ALPHA16I_EXT, 16, 256, 0,
              GL_ALPHA_INTEGER_EXT, GL_INT, triangleTableField);
```

Algoritam 5.3 Postavke teksture tablice trokuta

U polju algoritam 5.4 nalaze se postavke za teksturu podataka. Vrijednosti teksture se interpoliraju, a izvan područja teksture se također uzimaju najbliže vrijednosti. Interpolacija omogućuje crtanje više ćelija nego što ih je predviđeno podatkovnom rešetkom. U slučaju teksture podataka, pohranjuju se vrijednosti s posmačnim zarezom također u *alpha* komponenti boje. Vrijednosti `nx`, `ny` i `nz` su dimenzije podatkovnog polja.

```
glBindTexture(GL_TEXTURE_3D, dataTexture);
glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_3D, GL_TEXTURE_WRAP_R, GL_CLAMP_TO_EDGE);

glTexImage3D( GL_TEXTURE_3D, 0, GL_ALPHA32F_ARB, nx, ny, nz, 0,
GL_ALPHA, GL_FLOAT, dataField);
```

Algoritam 5.4 Postavke teksture podatkovnog polja

Polje točaka predstavlja ulaz u algoritam i koordinate pojedine točke se koriste za dohvat vrijednosti iz teksture podataka. To znači da je polje točaka pravilno raspoređena rešetka koja razapinje prostor valjanih koordinata teksture, od koordinate $(0,0,0)$ do koordinate $(1,1,1)$. Kako bi algoritam pokretnih kocaka crtao zatvorene površine na rubovima rešetke, potrebno je proširiti rešetku za jedno polje u svakoj dimenziji. Za početak definira se udaljenost između dviju susjedne pozicije:

$$(dx, dy, dz) = (\frac{1}{nx-1}, \frac{1}{ny-1}, \frac{1}{nz-1})$$

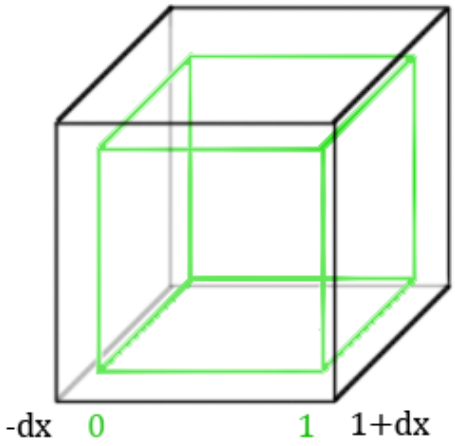
Zatim se izrađuje prostorna rešetka od koordinata $(-dx, -dy, -dz)$ do koordinata $(1,1,1)$ uz korak (dx, dy, dz) za svaku dimenziju.

Program za sjenčanje vrhova prosljeđuje vrhove kroz grafički cjevovod. Sve transformacije vrhova potrebno je raditi u sljedećem koraku grafičkog cjevovoda.

Unutar programa sa sjenčanje geometrije koji se provodi nad svakom točkom ranije definirane rešetke, radi se sljedeći postupak:

1. Određuju se koordinate vrhova ćelije. Konačna ćelija razapinje prostor između vrha (x, y, z) i vrha $(x + dx, y + dy, z + dz)$. Ovime se osigurava da ukupan prostor algoritma sa svih strana obuhvaća prostor koordinata tj. da se nalazi između točaka $(-dx, -dy, -dz)$ i $(1 + dx, 1 + dy, 1 + dz)$ kao što je ilustrirano slikom 5.3.
2. Određuju se vrijednosti rešetke iz teksture.

3. Računa se indeks kocke na temelju vrijednosti koje prelaze prag.
4. Interpoliraju se vrhovi ćelije i dobivaju se vrhovi izlaznih trokuta.
5. Temeljem tablice trokuta stvaraju se izlazne primitive čije se točke prethodno transformiraju matricama pogleda i projekcije.



Slika 5.3 Prostor koordinata (zeleno) i prostor obuhvaćen rešetkom

U programu za sjenčanje geometrije potrebno je definirati ulazne primitive kao točke, izlazne primitive kao trokute te postaviti maksimalan broj izlaznih točaka na 15. Definicija ulaznih parametara nalazi se u polju algoritam 5.5. Osim izlaznih primitiva definira se i izlaz `outPosition` koji predstavlja koordinate podatkovne teksture koje će biti potrebne u programu za sjenčanje fragmenata. Naime iz skalarnog polja podataka moguće je računati normalu na temelju gradijenta vrijednosti, a za to je potrebna pozicija fragmenta unutar podatkovnog polja. Ta pozicija ne smije biti interpolirana perspektivno, već linearno te se zato definira s ključnom riječi `noperspective`.

```
layout( points ) in;
layout( triangle_strip, max_vertices = 15 ) out;
invec3inPosition[];
noperspectiveoutvec4outPosition;
```

Algoritam 5.5 Isječak koda koji definira varijable programa za sjenčanje geometrije

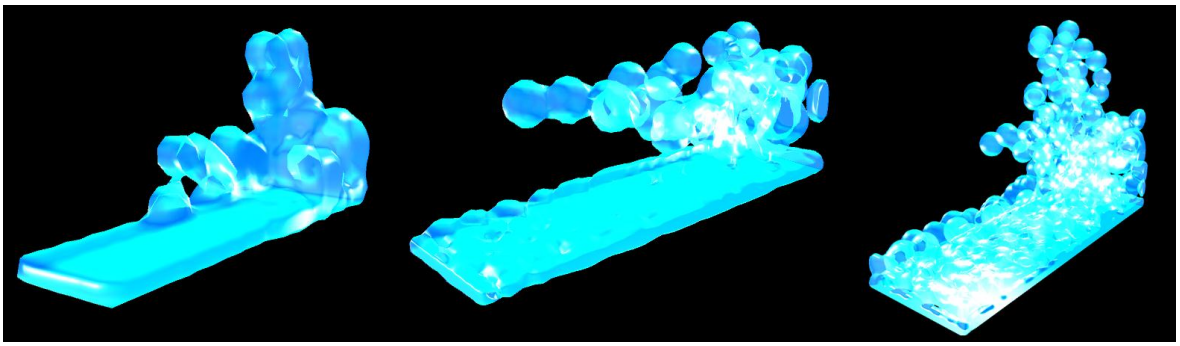
Unutar programa za sjenčanje fragmenata račun je proizvoljan. Računanje normale je jednostavno te je s njom moguće ostvariti primjerice osvjetljenje fluida. Uz korištenje miješanja alpha komponente jednostavno je ostvariti prozirni fluid, a korištenjem normale interesantna je ideja napraviti preslikavanje okoline na fluid.

Popis svih parametara potrebnih za program za sjenčanje dan je u tablici 5.4.

Tablica 5.4 Uniformni parametri programa za sjenčanje pokretnih kocki

Naziv	Tip	Opis
MVP	mat4	Matrica transformacije u prostor pogleda i projekcije
Data	sampler3D	Ulazni podatci iz kojih se stvara poligonalna mreža
DataStep	vec3	Razmak između susjednih točaka rešetke
TriTable	sampler2D	Tablica trokuta
Threshold	float	Vrijednost praga
VerticeOffsets[8]	vec3	Niz pomaka pomoću kojih se određuju vrhovi u ćelijama
Color	vec3	Boja fluida
Diffuse, Specular, Ambient	vec3	Boje pojedinih komponenti svjetla
LightPosition	vec3	Položaj svjetla

Konačno, fluid nacrtan programom za sjenčanje pokretnih kocki prikazan je na slici 5.5, a za usporedbu je dan prikaz za veličine rešetke 32^3, 64^3 i 128^3.



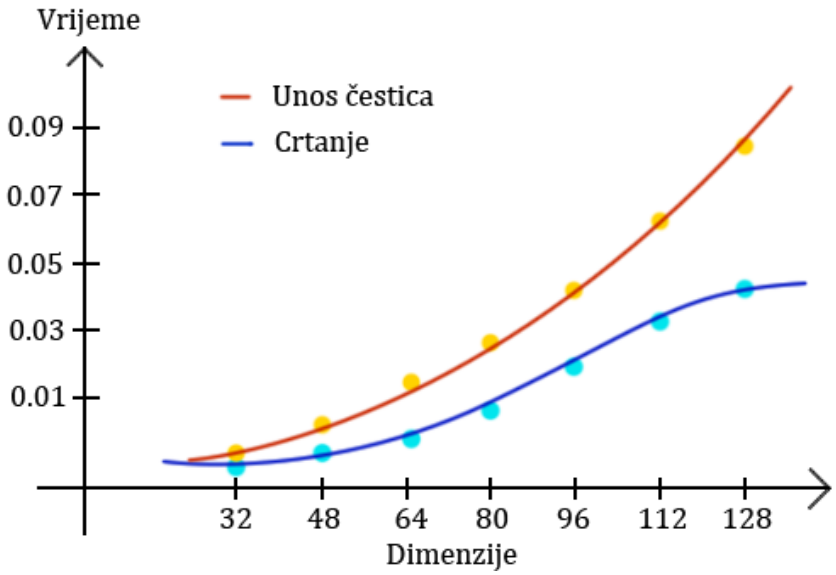
Slika 5.5 Pokretne kocke na GPU sa dimenzijom rešetke 32,64 i 128

U sustavu na slici je oko 350 čestica, a brzine izvođenja prikazane su u tablici 5.5. Za bolju usporedbu dane su i brzine za druge dimenzije rešetke. „Unos čestica“ je vrijeme potrebno za punjenje rešetke česticama. „Crtanje“ je trajanje crtanja postavljen rešetke. Sva su mjerenja prosjeku računata unutar 3 sekunde izvođenja te u trenutku dodavanja novih čestica kao na slici 5.5.

Tablica 5.5 Ovisnost vremena crtanja o dimenzijama rešetke

Dimenzije	32	48	64	80	96	112	128
Unos čestica	0.0025	0.0061	0.0135	0.0269	0.0379	0.0573	0.0858
Crtanje	0.0007	0.0015	0.0028	0.0083	0.0178	0.0304	0.0415

Graf dane ovisnosti vremena i dimenzija je prikazan na slici 5.6. Iz priloženog je očito da ovisnost postoji i da bi bilo potrebno poboljšati način unosa čestica u podatkovnu rešetku kako bi bilo moguće simulirati veći broj čestica.



Slika 5.6 Graf ovisnosti dimenzije rešetke i vremena crtanja

OpenGL napomene

S obzirom na to da je implementacija rađena u OpenGL-u 4, postoje neke funkcionalnosti koje su formalno izbačene iz te specifikacije, ali ih je moguće koristiti zbog održavanja kompatibilnosti sa starijim programima, izrade manjih i jednostavnijih programa te koji put čak i boljih performansi. Sve stare funkcionalnosti je moguće ostvariti u modu kompatibilnosti (*engl. compatibility mode*) dostupnim od OpenGL 3.2.

Crtanje primitiva je najjednostavnije pozivanjem funkcije `glBegin` s parametrom koji određuje što se crta te nizanjem točaka prostora koje će biti prikazane. Iako je ova metoda jednostavna i prihvatljiva kao početna točka u učenju rada s OpenGL-om, izbačena je zbog svoje sporosti i neprikladnosti za moderne grafičke sustave. Takozvani *immediate mode* crtanja je izbačen sa specifikacijom 3, kao i većina dijelova pripadnog sustava crtanja.

Mehanizam ubrzavanja crtanja prikaznom listom (*engl. display list*) je izbačen u OpenGL-u 3.2. Taj je postupak omogućavao pohranu djela koda koji vrši crtanje i njegovo ponovno pozivanje u bilo kojem trenutku uz prethodno optimiziranje tog poziva. Mehanizam je izbačen jer njegovo održavanje nije imalo smisla uz nove metode crtanja.

Crtanje se prema novim specifikacijama vrši pozivom funkcije `glDrawArrays` ili funkcije `glDrawElements`. Prvi način se koristi kada su definirani objekti nizova vrhova (VAO, *engl., vertex array object*) i objekti spremnika vrhova (VBO, *engl., vertex buffer object*) te kada ne postoji informacija o povezanosti primitiva, odnosno nije moguće

ponovno iskoristiti postojeće vrhove. Drugi način se koristi kada postoji informacija o povezanosti vrhova (npr. u obliku indeksa vrhova trokuta) namijenjenih za niz ili spremnik vrhova. Tada je pojedine vrhove moguće iskoristiti više puta i poziv `glDrawElements` troši manje memorije i daje bolje performanse. Prednost korištenja VAO je mogućnost definiranja i povezivanja VBO objekata koji predstavljaju različite podatke poput normala ili boja te pozivanje funkcije crtanja nad samo jednim VAO objektom koji objedinjuje sve podatke. Kako bi se u pozivu `glDrawElements` koristile boje i normale, potrebno je definirati zasebne spremnike i pozvati ih prije crtanja. Oba načina su pogodna za korištenje s programima za sjenčanje.

U najnovijim inačicama OpenGL-a izbačen je stog matrica. To znači da programer sam treba paziti na matrice koje se predaju u grafički cjevovod i da je potrebno računati matrice transformacija bez korištenja ugrađenih OpenGL funkcija. U modu kompatibilnosti stog je i dalje moguće koristiti, što može biti korisno u nedostatku matematičke biblioteke za rad s matricama iako nije preporučljivo.

Poanta svih promjena i izbacivanja funkcionalnosti iz OpenGL API-ja je zadržavanje veće kontrole nad tokom podataka, bolje upravljanje radom cjevovoda i transformacija programa u smjeru modernih grafičkih sustava koji koriste sjenčanje i arhitekturu koja dopušta protok velike količine podataka.

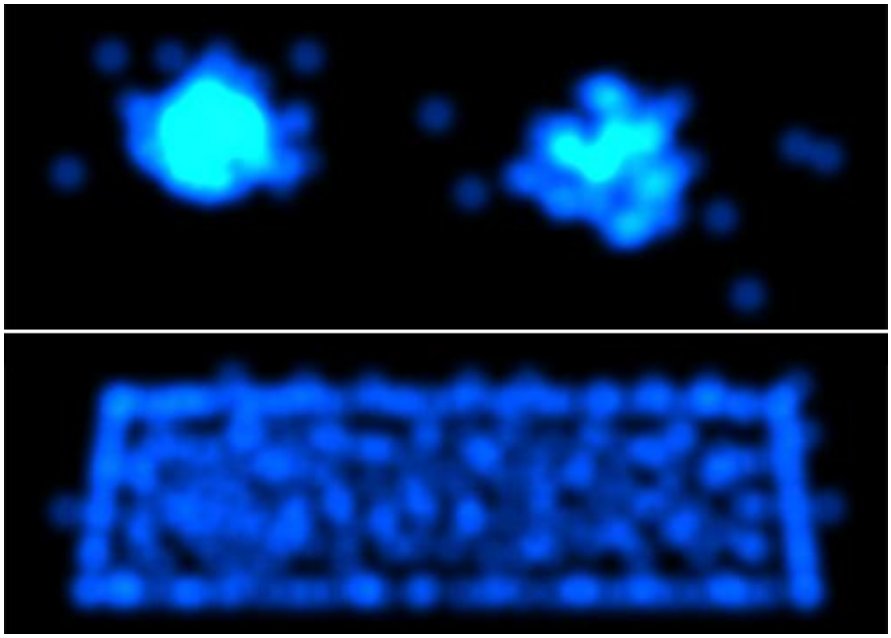
5.6 Problemi pri implementaciji

Tijekom razvoja implementacije javljali su se problemi s kojima se vjerojatno susreću svi programeri koji pokušavaju implementirati SPH sustav. Dokumentaciju tih problema je nažalost jako teško, gotovo nemoguće pronaći. Postoje mnoga pitanja oko izvedbe sustava i nekih problema te je koji put potrebno posegnuti za vlastitim rješenjima. U ovom potpoglavlju biti će navedene neke od problematičnih situacija s naglaskom na one češće i složenije.

Problemi unutar SPH sustava

Jezgrene funkcije, njihove derivacije, izrazi za silu i odnosi vektora su poprilično jasna stvar te ih nije teško implementirati niti razumjeti. Problem nastaje kada negdje dođe do pogreške u predznacima. Naime krivi predznak neće nužno uzrokovati potpunu promjenu ponašanja, ali će ta promjena biti dovoljna da fluid ne djeluje „ispravno“.

Primjerice, izrazi za gradijente jezgri sadrže negativan vektor položaja pomnožen s faktorom. Promatranjem samo faktora dobije se graf pozitivan na području definicije kao što je prikazan na slikama 3.2, 3.3 i 3.4 dok je stvarni iznos gradijenta negativan. Pogrešku s predznacima je vrlo jednostavno previdjeti na ovom mjestu, kao što prikazuje slika 5.7. Grupiranje čestica kao na gornjem djelu slike nastaje kada se postave krivi predznaci uz sile tlaka, dok je očekivani rezultat za usporedbu dan na donjoj slici.



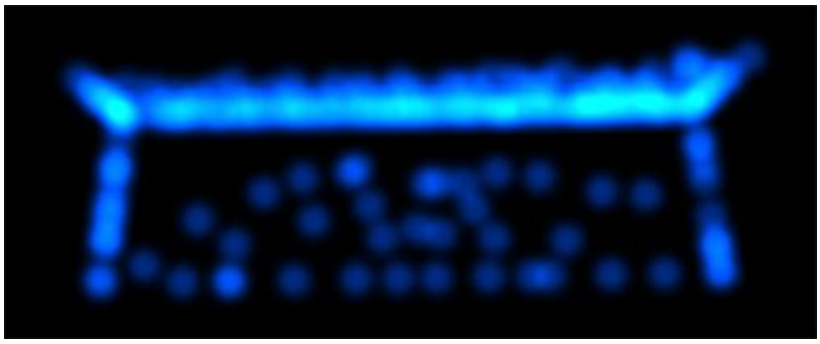
Slika 5.7 Razlika u ponašanju fluida kada su predznaci sila pogrešni (gore)

Iako čestice djeluju jedna na drugu odbojnim silama tlaka, zbog lošeg vremenskog koraka ili čistom slučajnošću, moguća je situacija iznimno bliskih čestica sličnih brzina i sila. Ako se dopusti velika blizina čestica, njihova udaljenost će biti bliska nuli. To znači da će u izrazu za silu tlaka koji koristi jednadžbu jezgre:

$$\nabla W_{spiky}(\vec{r}, h) = -\hat{\mathrm{r}}\frac{45}{\pi h^6}(\mathrm{h}-\mathrm{r})^2$$

doći do nedopuštenog iznosa sile. Naime normalizacija vektora je nestabilna za duljine vektora bliske nuli te su moguće nedefinirane ili beskonačne vrijednosti uzrokovane dijeljenjem s nulom. Rješenje je paziti da se čestice previše ne približe ili uvesti ograničenje na minimalnu udaljenost između čestica. Ako se uvede minimalna udaljenost, mogući su problemi konvergirajućih čestica. Naime za malene vrijednosti udaljenosti, iznos jezgre će biti malen, a s njime i sile. To će dovesti do ujednačavanja i grupiranja čestica što nije željeno ponašanje.

Površina interakcije bi trebala djelovati silom na čestice koje dolaze blizu nje i u najgorem slučaju izravno pomaknuti čestice u smjeru suprotnom od površine. Ako se čestice odbijaju bez prigušenja, događati će se puno nepotrebnog i nestvarnog skakutanja čestica po površini. Kada sile nisu dobro usmjerene (krivi predznak) ili ako njihov utjecaj nije dovoljno jak, moguće je da čestice zapnu u stabilnom stanju između djelovanja sile i odbijanja od površine. Ta je situacija češća uz presjecište dviju površina, kao što prikazuje slika 5.8 gdje su čestice grupirane uz rubove omeđujuće kocke zato što je smjer sile tlaka površine pogrešnog predznaka.



Slika 5.8 Grupiranje čestica uz površinu zbog krivog računa sila

Nažalost probleme u SPH sustavu nije lako uočiti, a još je teže pronaći njihov izvor zbog puno mjesta gdje je pogreška moguća. Na sreću, probleme nije previše lako uočiti te odstupanje od ponašanja neće uvijek biti u prvom planu pa barem vizualni dojam neće nužno biti upropašten.

OpenGL problemi

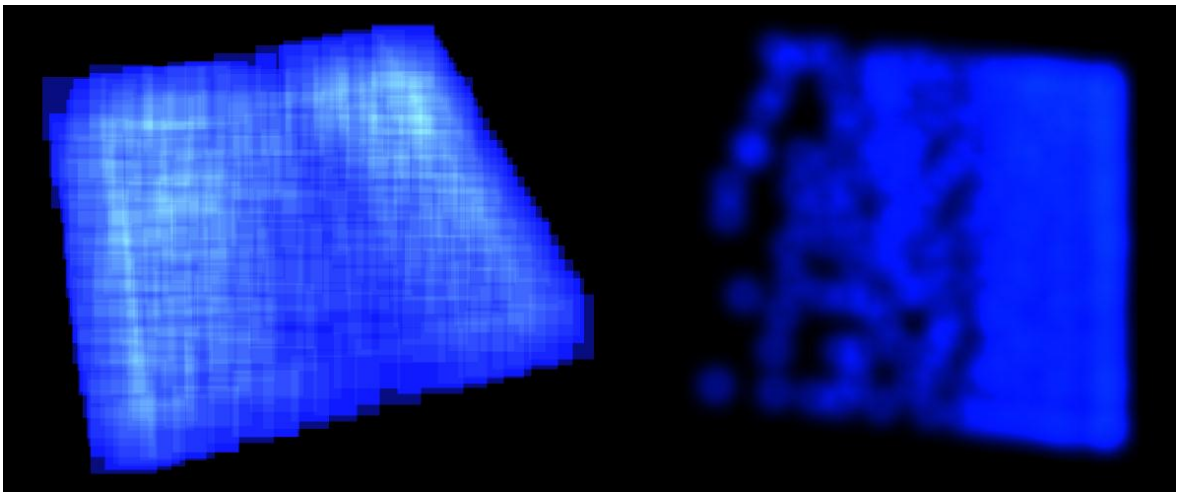
Ovi problemi prvenstveno nastaju kada je iskustvo u radu s OpenGL API-jem slabo i potrebno je proučiti što se i na koji način radi. Dodatan problem je činjenica da je specifikacija mijenjana te da je dopušteno koristiti stare funkcionalnosti iako su uklonjene.

U slučaju greške s korištenjem neke od funkcija crtanja teško je zaključiti gdje je greška nastala i kako ju ukloniti. Dobar primjer je korištenje ekstenzija koje omogućuju pristup funkcionalnostima grafičkog podsustava. Neke je funkcionalnosti moguće ostvariti tek ako se pozove funkcija s ispravnom ekstenzijom. Primjerice, za definiranje parametara točke za ugrađenu funkcionalnost crtanja polja sličica moguće je pozvati tri funkcije: `glPointParameterf`, `glPointParameterfARB` i `glPointParameterfEXT`.

Sufiksi ARB i EXT zapravo znače da su te funkcije odobrene od strane OpenGLrevizornog tijela (ARB, *engl., Architecture Revision Board*) odnosno da je dotična

ekstenzija prihvaćena od strane više proizvođača grafičkih kartica [33]. Različite bi ekstenzije mogle rezultirati različitim ponašanjem, što je moguće provjeriti jedino testiranjem.

Koliko je teško otkriti stvarni uzrok greške pokazuje primjer ponašanja implementacije uz korištenje ugrađenog prikaza polja sličica. Crtanje polja sličica bilo je neispravno kao što je prikazano na lijevom djelu slike 5.9. Na prvi mah pretpostavljeni je uzrok neispravnog ponašanja definicija krivih parametara i pozivanje krivih funkcija. U konačnici je otkriveno da je ranije u programa bila omogućena ručica za 3D teksturu (pozivom funkcije `glEnable(GL_TEXTURE_3D)`) te da nije bila pravovremeno uklonjena što je uzrokovalo probleme pri ugrađenom crtanju polja sličica.



Slika 5.9 Prikaz polja sličica je bio pogrešan zbog uključenih 3D tekstura

GLSL problemi

Slično kao i sa specifikacijom OpenGL-a, GLSL ima manu čestog mijenjanja što znači da najnoviji standard jezika za sjenčanje izbacuje mnoge funkcionalnosti prethodnih verzija i uvodi nove principe rada kako bi održao kompatibilnost s novim verzijama OpenGL-a. Primjerice, u verzijama do 1.4 transformacije u prostor pogleda i perspektivne transformacije su se vršile funkcijom `ftransform()`, koja je izbačena zajedno sa stogom matrica.

Razvoj programa za sjenčanje je moguć i u posebnim alatima, ali u onima dostupnima za razvoj GLSL programa, nije moguće razvijati programe za sjenčanje geometrije. Problem s pisanjem programa za sjenčanje u alatu kao što je Visual Studio, je taj što ne postoji ugrađena podrška za bojanje koda i automatsko nadopunjavanje

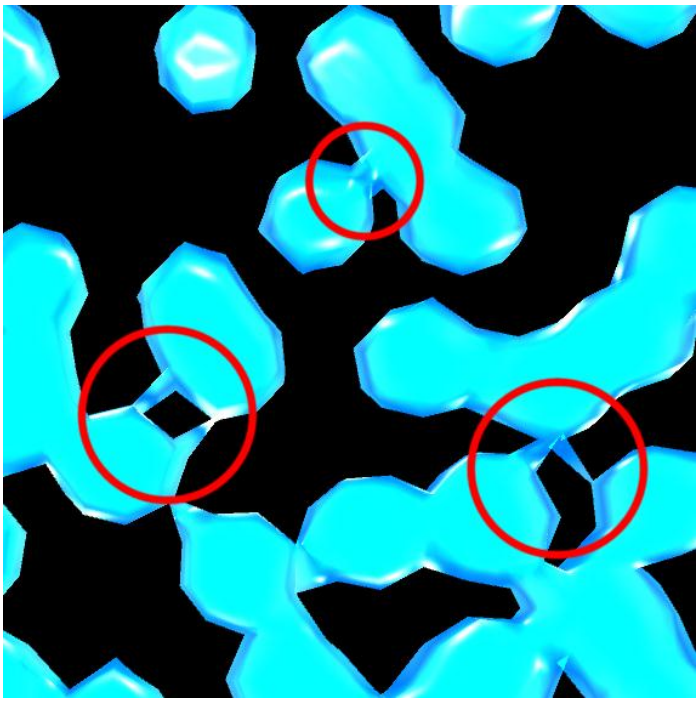
(*IntelliSense*). Djelomično rješenje je dodatak Nshader koji omogućuje bojanje koda unutar tog razvojnog okruženja.

Ostali problemi s GLSL-om su oni uobičajeni za razvoj programa za sjenčanje. Programiranje je koji put neuobičajeno zbog istovremeno niskog rada s jednostavnim elementima scene i prikazane visoke razine apstrakcije postupka. Teško je pronaći dobre upute i primjere za učenje koji prate razvoj specifikacije. Najveći je problem detekcija i otklanjanje logičkih pogrešaka jer ne postoji jednostavan način ispisa statusa programa. Potrebno je biti maštovit i bojama na ekranu dati do znanja što se točno događa. U razvoju implementacije pokretnih kocki dogodio se slučaj pogrešne logike programa i jedini način pronalaska greške bio je testiranjem pojedinih varijabli i crtanjem jednostavnih primitiva ako su varijable bile u određenom stanju.

Problemi pokretnih kocki

Dugačak je put do razumijevanja algoritma pokretnih kocki zbog puno raznih slučajeva i načina na koji je postupak moguće ostvariti. Dobar pristup je preko implementacije algoritma u dvije dimenzije proširiti princip na tri dimenzije. Problemi koji se javljaju su u orijentaciji trokuta, računanju i odabiru pravih slučajeva te implementaciji interpolacije. Iako je greška u postupku vidljiva, problem je otkriti što je točno uzrokovalo tu grešku. Moguće su greške s predznacima, krivim redoslijedom bridova u tablici ili krivim redoslijedom vrhova u trokutima. Iako su korištene unaprijed izračunate tablice preuzete s Interneta [34], ne postoji jamstvo da su tablice ispravne, a otkrivanje takve greške bilo bi iznimno teško.

Iako nije problem prilikom razvoja, osnovna varijanta algoritma može dati neočekivane rezultate prikaza za neke rubne slučajeve. Riječ je o situaciji kada su dvije čestice dovoljno blizu da između njih nastane veza, ali nisu dovoljno blizu da ta veza izgleda dobro. Slika 5.10 prikazuje taj slučaj, crveno su zaokruženi poligoni koji nisu potrebni kao „most“ između dviju čestica.



Slika 5.10 Artefakti u prikazu pokretni kockama

Problemi s C++ jezikom

U velikom projektu poput izrade SPH sustava moguće je naučiti puno novih i boljih načina korištenja jezika što stvara problem revizije starih dijelova sustava. Konzistencija u konvenciji pisanja koda jako je bitna, kao i pisanje komentara. Iako nije problem jezika, odstupanje od smjernica dobrog programiranja može olakšati stvaranje grešaka i znatno usporiti njihovo otklanjanje.

Konkretniji problemi s C++-om su oni vezani uz upravljanje memorijom. Ako upravljane nije dobro izvedeno može doći do usporavanja sustava posebice u SPH simulaciji koja koristi puno memorije. U slučaju propusta rada s memorijom, vrlo brzo bi moglo doći do pucanja programa.

Korištenje naprednih programskih koncepata jezika kao što su predlošci može djelovati kao dobra ideja za modularizaciju i u nekim je slučajevima isplativo. Primjerice za matematičke funkcije i podatkovne razrede vektora i matrice. Cijena je vrijeme učenja korištenja predložaka, kompliciran način prevođenja datoteka u kojima su deklarirani ti razredi i veća složenost koda.

Problemi mogu nastati i u pretjeranom optimiziranju, posebice po pitanju pristupa podatcima. Naime česticama SPH sustava je moguće pristupati indeksiranjem u nizu ili referenciranjem, što je bitno za izradu listi susjedstva. Ako se koristi indeksiranje pitanje

je brzine pristupa, a ako se koriste reference pitanje je isplativosti za veći broj čestica. Testiranje nije jednostavno i moguće je zapeti u odabiru i konačno loše dizajnirati sustav uz puno utrošenog vremena. Postoji još i odluka o tome kakav oblik pohrane čestica koristiti, niz ili strukturu *vector<T>* iz standardne biblioteke predložaka. Uobičajena je preporuka koristiti *vector* zbog ugrađenih funkcionalnosti, boljeg upravljanja prostorom pa čak i boljih performansi.

6 Zaključak

Problem simulacije fluida jedan je od težih i zahtjevnijih problema računalne grafike. Za njegovo razumijevanje potrebno je dobro znanje matematike i fizike te znanja iz područja računalne grafike kako bi fluid bio ispravno prikazan.

Iako je dostupno mnogo metoda simulacije temeljenih na prostornoj rešetci i na čestičnim sustavima, SPH metoda je jedna od najpopularnijih jer uz mnoge optimizacije postaje jak alat za simulaciju fluida u stvarnom vremenu. S obzirom na to da se ponašanje fluida u potpunosti definira njegovim parametrima, moguće su mnoge varijante fludia koje simuliraju ponašanja u različitim situacijama.

Prikaz fluida u realnom vremenu ima mnoge primjene, kao što je u medicini, biologiji, kemiji, agrokulturi i mnogim drugim prirodnim znanostima. Osim u znanstvene svrhe razvoj kvalitetnih SPH sustava bitan je i industriji zabave jer omogućuje stvaranje vjernijih efekata i bolje interakcije.

Budućnost razvoja SPH metode kreće se prema detaljnijem opisu fizikalnog modela fluida. Na području vizualizacije postoje mnoge nove metode koje treba testirati i istražiti za dodatna ubrzanja. Daljnji rad mogao bi istraživati korištenje SPH sustava za postizanje efekata kao što su miješanje fluida i utjecaj topline na fluid, trebao bi provjeriti i implementirati druge metode vizualizacije te optimizirati rad algoritma nekom od opisanih metoda.

Bruno Mikuš, lipanj 2012., Zagreb

_____

Literatura

[1] Gingold, R. A., Monaghan, J. J. Smoothed particle hydrodynamics: Theory and application to non-spherical stars. Monthly Notices of the Royal Astronomical Society, svezak 181, god. 1977., str. 375.-389.

[2] Lucy, L. B. A numerical approach to the testing of the fission hypothesis. Astronomical Journal, svezak. 82, god. 1977, str. 1013.-1024.

[3] Monaghan, J. J. Smoothed particle hydrodynamics. Annual review of astronomy and astrophysics, svezak 30, god. 1992., str. 543.-574.

[4] Stam, J., Fiume, E. Depicting fire and other gaseous phenomena using diffusion processes. Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, New York, USA, (1995.), str. 129.-136.

[5] Desbrun, M., Gascuel, M.-P. Smoothed particles: A new paradigm for animating highly deformable bodies. Computer Animation and Simulation, (1996.), str. 61.-76.

[6] Müller, M., Charypar, D., Gross, M. Particle-based fluid simulation for interactive applications. Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, Aire-la-Ville, Switzerland, (2003.), str. 154.-159.

[7] Müller, M., Solenthaler, B., Keiser, R., Gross, M. Particle-based fluid-fluid interaction. Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, New York, USA, (2005.), str. 237.-244.

[8] Stam, J. Stable Fluids. Proceedings of the 26th annual conference on Computer graphics and interactive techniques, New York, USA, (1999.), str. 121.-128.

[9] Foster, N., Fedkiw, R. Practical animation of liquids. Proceedings of the 28th annual conference on Computer graphics and interactive techniques, New York, USA, (2001.), str. 23.-30.

[10] Enright, D., Marschner, S., Fedkiw, R. Animation and rendering of complex water surfaces. Proceedings of the 29th annual conference on Computer graphics and interactive techniques, New York, USA, (2002.), str. 736.-744.

[11] Matsuda Y., Dobashi Y., Nishita T., Fluid simulation by particle level set method with an afficient dynamic array implementation on GPU, 2007.

[12] Koshizuka, S., Oka, Y. Moving-particle semi-implicit method for fragmentation of incompressible fluid. Nuclear science and engineering, svezak 123, god. 1996., str. 421.-434.

[13] Clavet, S., Beaudoin, P., Poulin, P. Particle-based viscoelastic fluid simulation. Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, New York, USA, (2005.), str. 219.-228.

[14] Dyken, C., Ziegler, G., Theobalt, C., Seidel, H.-P. High-speed Marching Cubes using HistoPyramids. Computer Graphics Forum, svezak 27, (2008.), str. 2028.–2039.

[15] Müller, M., Schirm, S., Duthaler, S. Screen space meshes. Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, Aire-la-Ville, Switzerland, (2007.), str. 9.-15.

[16] Kanamori, Y., Szego, Z., Nishita, T. GPU-based Fast Ray Casting for a Large Number of Metaballs. Computer Graphics Forum, svezak 27, (2008), str. 351.-360.

[17] Pfister, H., Zwicker, M., van Baar, J., Gross., M. Surfels: surface elements as rendering primitives. Proceedings of the 27th annual conference on Computer graphics and interactive techniques, New York, USA, (2000.), str. 335.-342.

[18] Yu, J., Turk, G. Reconstructing surfaces of particle-based fluids using anisotropic kernels. Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Aire-la-Ville, Switzerland, (2010.), str. 217.-225.

[19] Keenan Crane, Ignacio Llamas, Sarah Tariq. Real-Time Simulation and Rendering of 3D Fluids. Zadnja izmjena 20. 4. 2011, izdano 12. 8. 2007. *GPU Gems 3, chapter 30: Real-Time Simulation and Rendering of 3D Fluids.* http.developer.nvidia.com/GPUGems3/gpugems3_ch30.html. 13. 6. 2012.

[20] Goswami, P., Schlegel, P., Solenthaler, B., Pajarola, R. Interactive SPH Simulation and Rendering on the GPU. Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Aire-la-Ville, Switzerland, (2010.), str. 55.-64.

[21] Harada, T., Koshizuka, S., Kawaguchi, Y. Smoothed Particle Hydrodynamics on GPUs. Structure, 2008. Str. 1.-8.

[22] Morris, J. P. Simulating surface tension with smoothed particle hydrodynamics. International journal for numerical methods in fluids 2000., svezak 33., str. 333.-353.

[23] Glyde, G. *Example for Metaballs*, 03.01.2007. http://commons.wikimedia.org/wiki/File:Metaballs.png, 05.06.2012.

[24] GLEW, *OpenGL Extension Wrangler Library*, 15.04.2012., http://glew.sourceforge.net/

[25] SFML, *Simple and Fast Multimedia Library*, 15.04.2012., http://www.sfml-dev.org/

[26] DevIl, *Developers Image Library*, 15.04.2012., http://openil.sourceforge.net/

[27] NShader, 25.04.2012., http://nshader.codeplex.com/

[28] Teschner, M., Heidelberger, B., Müller, M., Pomerantes, D., and Gross, M. H. Optimized spatial hashing for collision detection of deformable objects. Vision Modeling and Visualisation (2003.), str. 47.-54.

[29] Onderik, J., Ďurkovič, R. Efficient Neighbourhood Search for Particle-based Fluids. Journal of the Applied Mathematics, Statistics and Informatics (JAMSI) 2007., svezak 3.

[30] Stroustrup, B. The C++ Programming Language, Third Edition. 1997.

[31] Lorensen, W., E., MarchingCubes: A High Resolution 3D Surface Construction Algorithm, Proceedings of the 14th annual conference on Computer graphics and interactive techniques SIGGRAPH, New York, USA, (1987.), str. 163.-169.

[32] Jean-Marie Favreau, MarchingCubes.svg, 16.10.2006., *Marching cube cases*, http://en.wikipedia.org/wiki/File:MarchingCubes.svg

[33] Mark J. Kilgard, *OGLextensions*, 10.6.2000., *All about Open GLextensions*, http://www.opengl.org/archives/resources/features/OGLextensions/

[34] PaulBourke, *Polygonise*, svibanj 1994., *Polygonising a scalar field*, http://paulbourke.net/geometry/polygonise/

Sažetak

Naslov: Animacija nestlačivih fluida temeljena na metodi SPH

U animiranoj grafici jedan od većih izazova je animacija fluida i postoje mnoge uspješne metode od kojih je među najpopularnijima metoda hidrodinamike izglađenih čestica. U ovom radu napravljen je pregled područja i radova koji su definirali i oblikovali SPH sustave. Dan je opis i pojašnjenje fizike fluida, razmotreni su efekti fluida poput viskoznosti i površinske napetosti te definirani pojmovi vezani uz SPH metodu kao što su jezgrene funkcije i izglađivanje svojstava. Navedene su neke česte metode prikaza fluida. Na kraju je opisana implementacija kompletnog sustava.

Ključne riječi: grafika, fluid, simulacija, animacija, hidrodinamika, čestice, izglađivanje svojstava, jezgrene funkcije, SPH, viskoznost, površinska napetost, OpenGL, GLSL, metakugle, polje sličica, pokretne kocke, sjenčanje geometrije.

Abstract

Title: Incompressible fluid animationbased on the SPH

One of the greater chalenges in animated graphics is the animation of fluids. Many succesfull methods of simulation exist of which smoothed particle hydrodynamics is probably the most popular one. This paper makes an overview of the field, with papers that defined and shaped current SPH systems. A description and an explanation of fluid physics is given, fluid effects as viscosity and surface tension have been considered and SPH method specifics such as kernel functions and attribute smoothing are defined. Several methods of fluid visualisation are also given. In the end an implementation of the complete system is given.

Keywords: graphics, fluid, simulation, animation, hydrodynamics, particles, smoothing, kernel, SPH, viscosity, surface tension, OpenGL, GLSL, metaballs, pointsprites, marchig cubes, geometry shading.