SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1236

SIMULACIJA FLUIDA SUSTAVOM ČESTICA

Marta Poštenjak

Zagreb, lipanj 2016.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 10. ožujka 2016.

Predmet: **Računalna grafika**

DIPLOMSKI ZADATAK br. 1236

Pristupnik: **Marta Poštenjak (0036465137)**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Simulacija fluida sustavom čestica**

Opis zadatka:

Proučiti fizikalne osnove dinamike fluida. Razraditi osnovne matematičke komponente u opisu toka fluida te proučiti mogućnosti implementacije na računalu. Posebice obratiti pažnju na modele temeljene na česticama. Proučiti postupke vizualizacije koji omogućuju prikaz dobivenih rezultata. Na različitim primjerima prikazati ostvarene rezultate. Diskutirati utjecaj različitih parametara. Načiniti ocjenu rezultata i implementiranog algoritama.
Izraditi odgovarajući programski proizvod. Koristiti programski jezik C++ i grafičko programsko sučelje OpenGL. Rezultate rada načiniti dostupne putem Interneta. Radu priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Zadatak uručen pristupniku: 18. ožujka 2016.
Rok za predaju rada: 1. srpnja 2016.

Mentor:

Prof. dr. sc. Željka Mihajlović

Djelovođa:

Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za diplomski rad profila:

Prof. dr. sc. Siniša Srbljić

Zahvaljujem se svojim roditeljima, teti Alemki, i sestrama.

Zahvaljujem se i profesorici Željki Mihajlović na mentorstvu i pomoći tijekom studiranja.

Sadržaj

Popis slika

Uvod

Upita li se bilo kojeg stručnjaka koji se bavi računalnom grafikom što je to najteže simulirati i prikazati na računalu, svi odgovori mogli bi se svesti na jedan zajednički nazivnik – fizikalno temeljena simulacija. Kolizija pramena kose s podlogom, turbulentni vrtlog, elastičnost pojedine tkanine, gravitacijske sile sustava zvijezda samo su neki od primjera. Stoga nije neobično da je i simulacija fluida zauzela mjesto pri vrhu ljestvice. Njezina problematika je dvojaka. Prvo, mehanika fluida sama po sebi traži veliko razumijevanje i poznavanje osnovnih teorijskih koncepata i načela, prepoznavanje i identificiranje različitih parametara, njihovog odnosa te načina na koji oni utječu na fluid. Potrebno je napraviti dobar model sustava. Tek onda može se razmišljati o prikazivanju navedenoga na računalu. Kad se to u konačnici i uspije svesti na matematički model i implementirati na računalu, dolazi se do drugoga problema. Računanje je vremenski iznimno skupo i zahtjevno. Ne samo da je potrebno napraviti ogroman broj izračuna prije iscrtavanja na ekran nego su tekstura i površina fluida problemi sami za sebe.

Tema ovoga diplomskoga rada bit će simulacija fluida s naglaskom na simulaciju sustavom čestica. Simulacija fluida grana je računalne grafike s ulogom prikazivanja fluida na ekranu. Proučit će se osnove simulacije te različiti pogledi na to. Napraviti će se implementacija nekih osnovnih metoda te usporediti dobiveni rezultati. Pokušat će se napraviti gradacija pregleda različitih tehnika počevši od najjednostavnije pa do one koja u sebi uključuje najviše fizike. Bez poznavanja osnovna fluida bilo bi neizvedivo simulirati njihovo ponašanje. Zato će se prvo poglavlje pozabaviti fizikalnom pozadinom i osnovnim svojstava fluida. Uslijedit će kratka povijest računalne dinamike fluida (CFD) te par riječi o tome kako se ona može iskoristiti za simulaciju fluida. S obzirom da će sve implementacije biti na programskom jeziku C++ uz OpenGL, opisat će se pojedinosti OpenGL-a i osnovna struktura programa koja će svoju formu zadržati u svim projektima. Zatim će se opisati nekolika različitih načina promatranja fluida, počevši od najjednostavnijeg ka složenijima. Poseban naglasak dat će se metodama koje fluid promatraju kao sustav čestica. Prva metoda koja će biti implementirana krajnje je jednostavna simulacija fluida česticama koja ne uzima u obzir interakcije među pojedinim

česticama. Zatim će se korištenjem proceduralne animacije i visinskih mapa generirati površina fluida. U zadnjem poglavlju pokušat će se u 2D prostoru implementirati metoda koja svoje korijene vuče iz CFD-a – jednostavni SPH simulator. Svi nepoznati pojmovi bit će bolje objašnjeni u daljnjem tekstu.

1. Osnovna svojstva fluida

Većina ljudi pojam fluida asocira s tekućinom. Međutim fluid je puno više od toga. Zrak koji nas okružuje, magma u vulkanima pa čak i zrnca od kojih se sastoje Saturnovi prstenovi spadaju u tu kategoriju. Ako pak proširimo kategoriju s tekućine na tvari u tekućem i plinovitom agregatnom stanju, kako onda objašnjavamo da i plastika često pokazuje svojstva fluida? Što je onda fluid zapravo? „Fluid je supstanca koja se kontinuirano deformira pod djelovanjem smicanja“. (McDonough, 2009) Smicanje je sila koja je paralelna sa smjerom gibanja (toka). Prethodna definicija poprilično dobro hvata srž, a to je činjenica da je fluid pod djelovanjem sile smicanja teče.

Fluidima se bavi zasebna disciplina fizike – mehanika fluida. Njena osnovna podjela je na hidromehaniku (kretanje fluida) i hidrostatiku (fluidi u mirovanju). Obje discipline svoje korijene sežu još u doba antičke Grčke i Arhimeda. Postoji nekoliko osnovnih zakonitosti koje su uvijek zadovoljene, to su:

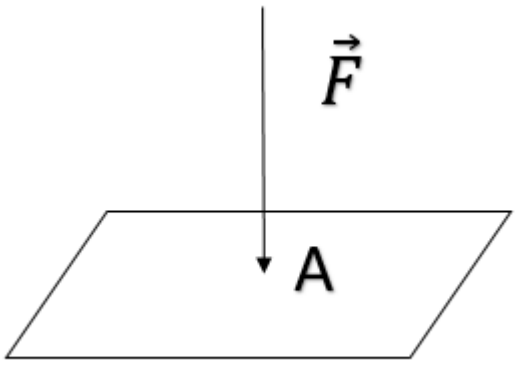
- Hipoteza kontinuuma
- Očuvanje mase
- Očuvanje gibanja
- Očuvanje energije

Hipoteza kontinuuma u kontekstu dinamike fluida znači sljedeće „Bez obzira koliko mali dio volumena fluida promatrali, on zadržava sva makroskopska svojstva fluida koja povezujemo s većom količinom“ (McDonough, 2009). Objašnjene očuvanja mase poprilični je trivijalno – bez obzira giba li se fluid ili miruje, ukupna masa ostaje nepromijenjena. Očuvanje gibanja reformulacija je drugog Newtonovog zakona, dok je očuvanje energije posljedica prvog zakona termodinamike. On kaže da se energija ne može uništiti ili stvoriti iz ničega. Može se samo prenijeti iz jednog oblika u drugi.

Osnovna svojstva fluida koja će direktno utjecati na njegovo gibanje bit će tlak, gustoća i viskoznost. Zbog svoje važnosti svaki će biti promotren u različitom potpoglavlju.

1.1. Tlak

Slika 1.1 Sila na jedinicu površine



“Tlak u fluidu nastaje uslijed njegove vlastite težine ili vanjskih sila, a karakteristično za njega je da djeluje uvijek okomito na plohu na kojoj ga se promatra.” (Andreić, 2014) Odgovora sili po jedinici površine (Slika 1.1).

$$p = \frac{F}{A}$$

Mjerna jedinica za tlak je Pascal (Pa) te odgovara sili od jednog N koja djeluje na površinu $1\ m^2$. Tlak je veličina koja se povećava s dubinom fluida jer viši slojevi svojom težinom pritiskuju donje slojeve. Također, na mjestima iste dubine iznos tlaka je jednak. Zamislimo neki tekući fluid koji miruje u posudi. Tlak kojim djeluje na dno posude ovisi samo o visini stupca te gustoći tekućine. Nikako ne ovisi o obliku posude u koju smo ga stavili ili ukupnoj količini tekućine u toj posudi. Ova pojava naziva se „hidrostatski paradoks“. Tlak je veličina koja se propagira kroz fluid trenutno te djeluje u svim smjerovima jednako. Bilo kakva promjena na jednom kraju istodobno će se osjetiti na drugom. Ta zakonitost je osnova rada hidrauličke dizalice koja koristi malu silu na malu površinu kako bi podigla objekt veće težine koristeći veću površinu.

1.2. Gustoća

Gustoća materijala svojstvo je neovisno o veličini promatranoga uzorka te se fizikalno opisuje kao koeficijent mase i volumena. Ukoliko je taj koeficijent stalan u svakom djeliću materijala, nazivamo ga homogenim. Kod nehomogenih materijala definicija gustoće se proširuje na promjenu mase po jedinci volumena.

$$\mathrm{D} = \frac{\mathrm{dm}}{\mathrm{dV}}$$

Fizikalna veličina za gustoću po SI sustavu je $\frac{kg}{m^3}$. To je izvedena mjerna jedinica koja proizlazi direktno iz formule. Gustoća je svojstvo koje uvelike ovisi o temperaturi. Osim kod vode, kod svih materijala opada s porastom temperature. Voda je najgušća na 4 °C. To je ključ života u vodi za vrijeme zimskih mjeseci. Led ja znatno lakši od tekuće vode pri niskim temperaturama. Kopnene vode se zbog toga lede od površine prema dubini.

Osim temperature, tlak također utječe na gustoću fluida. Kod tekućina to nije izraženo te je ta vrijednost neznatna. S druge strane, kod plinova tlak igra jako veliku ulogu te je povećanje tlaka proporcionalno povećavanju gustoće.

1.3. Viskoznost

Viskoznost fluida mjera je kojom se fluid opire deformaciji pri smicanju. To je zapravo koeficijent unutarnjeg trenja fluida. Matematički opis ovoga svojstva je koeficijent smicanja i gradijenta brzine.

$$\mu = \frac{\frac{F}{A}}{\frac{du}{dy}}$$

Ova formula odgovara dinamičkoj viskoznosti. Postoji još i kinetička viskoznost. Ona opisuje otpor uz gravitaciju i jednaka je koeficijentu dinamičke viskoznosti I gustoće materijala.

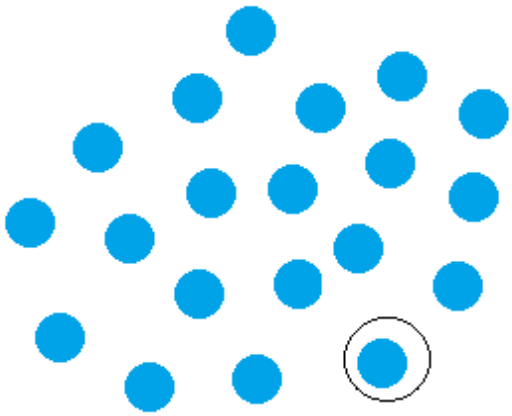
$$\nu = \frac{\mu}{\rho}$$

Uzroci viskoznosti kod plinova i tekućina su različiti. Kod prvih se javlja zbog molekularnih sila, dok kod potonjih difuzijske sile među slojevima uzrokuju izmjenu impulsa. Viskoznost je isključivo svojstvo materijala i javlja se samo pri gibanju. Što je veća, fluid se teže giba i obrnuto. Sada lako možemo zaključiti da je med puno viskozniji od vode koja je pak viskoznija od zraka (Slika 1.1). Nadalje, ona ovisi o temperaturi. Kod tekućina je obrnuto proporcionalna porastu temperature. Što je temperatura veća, molekule se sve više udaljavaju i manje utječu jedna na drugu. Kod plinova je drukčija situacija. Porast temperature uzrokuje sve kaotičnije gibanje molekula, što rezultira sve češćim kolizijama.

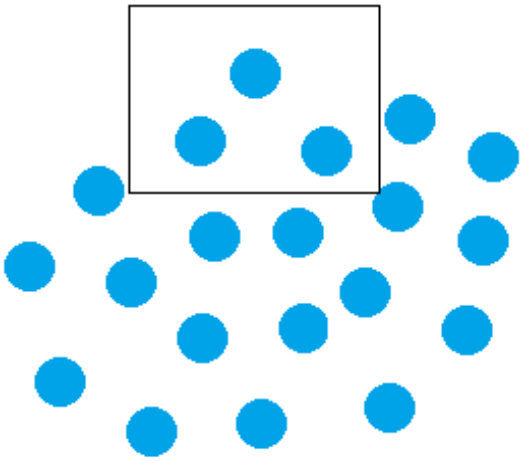
Ukoliko je viskoznost fluida kao koeficijent već navedenih veličina linearna funkcija, takav fluid smatrat ćemo njutnovski. Njutnovski fluid zadržava svojstva fluida bez obzira koliko ga miješali, trgali ili primjenjivali smicanje. Fluid je nenjutnovski inače. Kod takvog može doći do trganja i pojavljivanja rupa.

Slika 1.2 Med kao primjer viskoznog fluida (slika preuzeta iz Agricultural Research Service)

1.4. Osnovni pristupi razmatanju fluida



Slika 1.3 Lagrangeov način promatranja fluida



Slika 1.4 Eulerov način promatranja fluida

U fizikalnom razmatranju fluida dva su osnovna pristupa – lagrangeovski i eulerovski. Kako bi se bolje razumio pojedini pristup, povući će se analogija s restoranom. Zamislimo restoran s konobarima koji poslužuju po cijelom restoranu i stolovima kao fiksnim točkama. Serviranje hrane možemo promatrati na dva načina.

Prvi bi bio da pratimo svakog konobara i gledamo kuda se kreće i koje stolove obilazi. Jedan konobar će u jednoj večeri obići puno različitih stolova. Drugi bi bio da se fiksiramo na stolove i gledamo koji konobar dolazi do njega. U jednoj večeri puno različitih konobara će doći do stola. Prvi način odgovarao bi lagrangeovom, a drugi eulerovskom pristupu. Sada ćemo to primijeniti na fluide. Slike 1.3 i 1.4 predstavljaju fluid. Ukoliko ih promatramo lagranegovski (Slika 1.3) , fokusirat ćemo se na sitne dijelove fluida, tzv. čestice i pratiti njihovo kretanje u vremenskom kontekstu. Zasebno promatramo svaku plavu točkicu (česticu) sa slike. Svaka čestica nositelj je svojstva fluida kao što su masa, brzina... Svako takvo svojstvo čestice označava se u vremenskom kontekstu, npr v(t), x(t). Eulerovski pristup fiksirat će određene točke prostora i gledati što se događa u njima (Slika 1.4), što utječe, što istječe. Na slici 1.4 kvadrat označava fiksan prostor koji gledamo, a plave točkice su dio fluida koji se u promatranom trenutku tamo zatekao. Svako svojstvo označavat će se i u vremenskom i u prostornom kontekstu, primjerice v(x,t). Prostor se dijeli na imaginarnu rešetke i u vremenskom trenutku promatraju se stanja pojedinih ćelija. I jedan i drugi pristup imaju prednosti i mane. Različite fizikalne veličine bolje odgovaraju jednom ili drugom sustavu. Primjerice, kada pričamo o akceleraciji, puno ju je lakše i intuitivnije opisati u lagrangeovom sustavu. To je promjena brzine jedine čestice po jedinci vremena. Ako pak to pokušamo prenijeti u eulerov sustav, formula se znatno komplicira. U jednom vremenskom trenutku t_1 promatranjem određene ćelije i čestice u njoj. U vremenu t_2 te čestice više nisu tamo. One se sada nalaze na novoj poziciji koju također moramo uzeti u obzir pri računanju. Druge veličine poput viskoznosti, lakše je opisati u eulerovom sustavu. Iz ovoga proizlazi jedan od fundamentalnih problema dinamike fluida – različiti parametri bolje odgovaraju jednoj ili drugoj interpretaciji, ali većinom svi moraju biti uzeti u obzir. Moramo se odlučiti za jednu, ali koju god odabrali, uvijek će se morati žrtvovati nešto.

Ipak postoji način da se ova dva pristupa povežu, a to je materijalna derivacija. „Materijalna derivacija izražava brzinu promjene fizikalnog svojstva čestice fluida, tj. promjenu koju bi osjetio promatrač kad bi se gibao zajedno sa česticom“ (Šavar, Virag, & Džijan, 2014). Inicijalno se materijalna derivacija povezuje s lagrangeovim pristupom. Njenim zapisivanjem u eulerovom dobiva se direktna povezanost promjene svojstava eulerovom. Time će se primjerice bez problema moći zapisati

akceleracija u eulerovom sustavu koju smo već istaknuli kao kompliciranu komponentu.

$$\frac{DF}{Dt} = \frac{\partial F}{\partial t} + (u \cdot \nabla) F$$

Lijeva strana jednadžbe je lagrangeova dok je prvi izraz desne strane Eulerova interpretacija. Desna strana kaže da se promjena svojstva može događati iz dva razloga. Ili se svojstvo mijenja u vremenu (prvi član) ili je čestica otišla na mjesto gdje je drukčija vrijednost toga svojstva (drugi član). Sada se konačno može i raspisati izraz za akceleraciju u Eulerovom sustavu. U gornjoj formuli svojstvo F zamijeniti će se sa brzinom, koja se u mehanici fluida standardno označava sa slovom *u*, te se dobiva

$$\frac{Du}{Dt} = \frac{\partial u}{\partial t} + (u \cdot \nabla) u$$

U trodimenzionalnom prostoru brzina u sastoji se od triju komponenti: x, y, i z, od kojih je svaka usmjerena u odgovarajućem smjeru pripadajuće koordinatne osi. Simbol ∇ označava del operator i predstavlja parcijalnu derivaciju. Raspisivanjem parcijalne derivacije dolazi se do sljedećega oblika zapisa

$$\frac{Du}{Dt} = \frac{\partial u}{\partial t} + v_x \frac{\partial u}{\partial t} + v_y \frac{\partial u}{\partial t} + v_z \frac{\partial u}{\partial t}$$

1.5. Navier-Stokesove jednadžbe

Do početka 19. stoljeća za opis toka fluida koristila se Eulerova jednadžba.

$$\frac{D\vec{u}}{dt} = \vec{q} - \frac{1}{\rho} \nabla_p$$

Claude-Louis Navier, francuski fizičar s početka 19. stoljeća, uveo je viskoznost u tu jednadžbu. Otprilike 100 godina kasnije irski fizičar Gabriel Stokes redefinirao je te jednadžbe koje danas znamo pod imenom Navier-Stokesove jednadžbe, a čine fundamentalnu osnovu toka fluida. U kontekstu ovoga rada promatrat će se samo jednadžbe koje se odnose na nestlačive fluide. Nestlačivi fluid je onaj čiji je volumen postojan bez obzira na tlak i temperaturu. U realnosti takav fluid ne postoji. Kada bi

voda bila takav fluid, nikad ne bismo mogli čuti pod vodom. Ipak, tekućine se približavaju onome što bismo nazvali nestlačivim fluidom.

Jednadžbe svoje podrijetlo vuku iz drugog Newtonovog zakona F = m*$\vec{a}$, ali dodaju i dodatne parametre kao što je viskoznost. Prva jednadžba odnosi se na očuvanje mase i kaže da se prenošenjem mase vektorom brzine ukupna masa fluida ne može promijeniti.

$$\nabla \cdot u = 0$$

Druga jednadžba glasi

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = \vec{g} - \frac{1}{\rho} \nabla \vec{p} + \frac{\mu}{\rho} \nabla^2 \vec{u}$$

Na prvi pogled, jednadžba se čini kompliciranom. Zapravo se već upoznalo s njenim dijelovima u prethodnim poglavljima. Desna strana označava izraz za akceleraciju u Eulerovom pogledu na fluide. Ova jednadžba govori čemu je zapravo jednaka akceleracija, odnosno promjena brzine. Lijeva strana se sastoji od triju komponenti. Prva označava gravitaciju. Ukoliko su u sustav uključene neke dodatne sile koje djeluju na fluid, one će se također uključiti i ovaj dio po formuli

$$\vec{a} = \frac{F}{m}$$

Drugi dio odnosi se na tlak kojim čestice fluida djeluju jedna na drugu. Oko čestice javljaju se različita polja tlakova koja nisu istoga iznosa. Čestica se onda giba iz smjera višega prema smjeru nižega tlaka, suprotno od gradijenta vektora tlaka.

Zadnji dio desne strane odnosi se na viskoznost koju smo već definirali kao otpor fluida prema tečenju. Što je veća viskoznost, dio fluida koji se prvi počinje gibati jače će za sobom povlačiti ostatak fluida jer će se fluid više opirati toku. Viskoznost je usko povezana s brzinom te se nikad ne očituje, osim pri gibanju. To objašnjava komponentu brzine u jednadžbi.

Sada kada su poznate sve komponente jednadžbe, možemo je izreći riječima. Fluid se giba pod utjecajem gravitacijske sile, vanjskih sila koje djeluju na sustav, razlike tlakova i viskoznosti kao svojstva fluida.

Potrebno je definirati rubne uvjete Navier-Stokesovih jednadžbi. Fluid može imati dvije granice – podlogu i slobodnu površinu. Ako se nalazi uz podlogu brzina toga dijela fluida postavlja se na brzinu te podloge. Ako se podloga giba, daje se ta brzina, ako se ne giba stavlja se na nulu. Slobodna površina označava granicu s drugim fluidom kao što je zrak ili neka druga fluidna supstanca. U tom slučaju za drugi fluid treba definirati gustoću i brzinu granice.

2. Računalna dinamika fluida

2.1. Kratka povijest računalne dinamike fluida

„Uz danu godišnju količinu padalina, kako se moraju oblikovati odvodi kako bi uklonili pravu količinu vode?“ (Ashford, 1985) jedno je od najvažnijih pitanja povijesti računalne dinamike fluida (u daljnjem tekstu CFD). To je pitanje koje je postavljeno početkom 20. stoljeća mladom Lewisu Fry Richardsonu, eklektičnom engleskom znanstveniku široka polja interesa. Možda je baš to bila prva domina u domino efektu otkrića koji su CFD doveli do današnje razine. Ono što je on tada učio bilo je revolucionarno do te razine da su mnoge ozbiljne ustanove ideju odbacile kao previše aproksimativnu i smiješnu. Richardson je napravio slijedeće; imao je Laplaceovu jednadžbu, ali domena na kojoj je morao provesti izračun bila je presložena. Njegova ideja bila je upotreba aproksimativnih jednadžbi na samo malenim djelićima toka, gdje bi onda mogao koristiti osnovne matematičke zakonitosti. Odlika koja ga je istakla bila je promatranje problema koji se do tada nisu smatrali matematičkima u matematičkome svjetlu. Njegov doprinos nije se ograničio na samo jedno polje znanosti. Ostavio je neizbrisiv trag i u meteorologiji i psihologiji. Također, poznat je po velikom broju eksperimenata. Najpoznatiji je, zasigurno, pokušaj da se promatranjem vremenskih uvjeta jednoga dana predvidi vremenska prognoza za sutrašnji dan. Malo je reći da je procjena bila pogrešna. Međutim, danas je pokazano da bi se, primjenom metoda zaglađivanja na izračun, tadašnji izračun pokazao točnim.

Slijedeće veliki korak donio je trio znanstvenika Courant, Friedrichs i Levy. Oni su tridesetih godina proučavali linearne parcijalne diferencijalne jednadžbe. Pokazali su nužan uvjet koji mora biti zadovoljen da bi se takva jednadžba s diferencijala mogla svesti na razliku. Takav se postupak naziva metoda konačne razlike.

$$f'(x) = \frac{f(x + dx) - f(x)}{dx}$$

Uvjet koji mora biti zadovoljen kako bi se ta metoda mogla primijeniti naziva se CFL uvjet po inicijalima znanstvenika koji su ga otkrili.

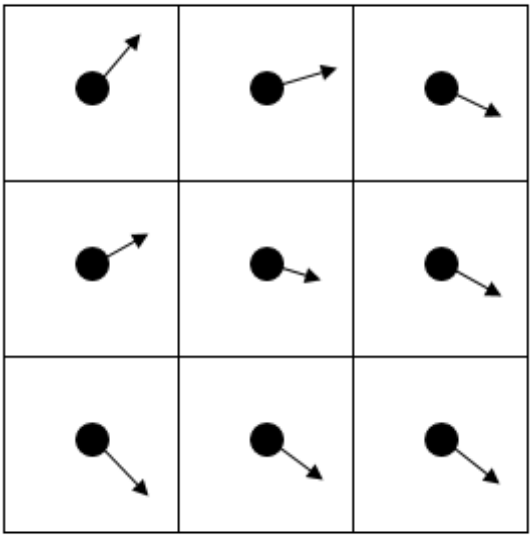
Šezdesetih godina znanstvenici Harlow i Welch dugotrajnim radom došli su do metode „Marker i ćelija“ (MAC). „Zasnovana na Euerovoj rešetci, MAC metoda je tehnika konačnih razlika za istraživanje dinamike nestlačivih viskoznih fluida (McKee, i dr., 2007). U osnovi algoritma bile su lagrangeove čestice koje su putovale s poljima brzine. One su označavala rubove granica fluida i fluida ili fluida i zraka. Ukoliko je neka ćelija sadržavala takvu česticu, bila je označena kao rub.

Metoda konačnih volumena javila se krajem stoljeća. Prostor se diskretizirao na malene volumene. U centru svakog takvog volumena bio bi nositelj podataka za taj volumen. Podjelom prostora moguće je diferencijalne jednadžbe svesti na algebarske.

2.2. Kratak pregled danas najzastupljenijih metoda

Danas najzastupljenije metode se slično kao i kod mehanike fluida dijele na dvije osnovne – one zasnovane na eulerovom i lagrangianovom pogledu na fluide. Obje imaju svoje prednosti i mane. Eulerov pogled često daje sporije i točnije rezultate, ali je lošiji kod prikazivanja turbulentnijih tokova kao što su zapljuskivanje ili eksplozija. Lagrangeov pristup je često brži, ali generalno manje precizan te ostavlja problem prikazivanja površine. Ukoliko bismo promatrali određen broj fiksnih točaka u eulerovom sustava te broj čestica u lagrangeovom, veća rezolucija bi se dobila u prvom. U svojoj osnovi i jedna i druga metoda sadrže rješavanje diferencijalne Navier-Stokesove jednadžbe.

2.2.1. Metode zasnovane na rešetci (eulerov pristup)



Slika 2.1 Prikaz vektora brzina fluida na rešetci

Eulerovske metode, slično kao i eulereov pristup promatranju fluida, diskretiziraju domenu na tzv. rešetku te promatraju što se događa u pojedinim ćelijama. Na takvoj domeni za dovoljno malu veličinu ćelije moguće je diferencijalne jednadžbe svesti na razliku. Svaka ćelija pamti za sebe podatke o dijelu fluida koji prolazi kroz nju (gustoća, tlak, brzina). Moguće je sve podatke pamtiti u istu rešetku, ali puno bolji pristup je raspoređena rešetka. „Rešetka se zove raspoređena zato što pamti različite veličine na različitim lokacijama.“ (Tech, Sandu, & Braley) Na početku algoritam inicijalizira se reštka te se fluid rasporedi po njoj. U svakom prolazu algoritma izračunavaju se nove vrijednosti svojstava pojedine ćelije u ovisnosti o brzini fluida i toj ćeliji i o prethodnim vrijednostima svojstava (Slika 2.1). Osnova svakog koraka je advekcija – prenošenje svojstava vektorom brzine. Opis advekcije je slijedeći: „Ako je Q neka količina na simulacijskoj rešetci, kako će se Q promijeniti Δt kasnije“. (Tech, Sandu, & Braley) Postoji više načina izračuna, od kojih nisu svi jednako stabilni.

Ovaj pristup ima svojih prednosti i mana. Ukupna masa nije uvijek očuvana. Potrebno je provoditi izračune te uključiti u rešetku svaki dio prostora u kojem se fluid može naći tijekom simulacije. Eulerove metode su stoga znatno sporije. Prednost im je preciznost simulacije te puno bolje opisivanje površine.

2.2.2. Metode zasnovane na česticama (Lagrangeov pristup)

Metode zasnovane na česticama temelje se na lagrangevom pogledu na fluide. Fluid je predstavljen skupom čestica te se računaju interakcije među njima. Osnovna prednost ovoga pristupa je brzina. Moguće je postići dobre rezultate u realnom vremenu te je pogodna za računalne igrice. Također, očuvanje mase je trivijalno jer konstantno baratamo s istim brojem čestica koje imaju jednaku masu cijelo vrijeme. Predstavnik ove skupine metoda, koji je danas najzastupljeniji, je metoda SPH (zaglađena hidrodinamika čestica), o kojoj će biti više riječi u zadnjem poglavlju. Glavni nedostatak je problem prikazivanja površine za koji je potrebno upotrijebiti neke dodatne tehnike. Jedna od najpoznatijih je tehnika marširajućih kocki (engl. *Marching cubes algorithm*). Njegovi kreatori su Cline i Lorensen. Do 2005. godine bio je zaštićen patentom, ali sada se smije slobodno koristiti. Algoritam za 8 susjednih točaka računa poligon koji bi najbolje predstavio površinu unutar te kocke u ovisnosti i položajima čestica.

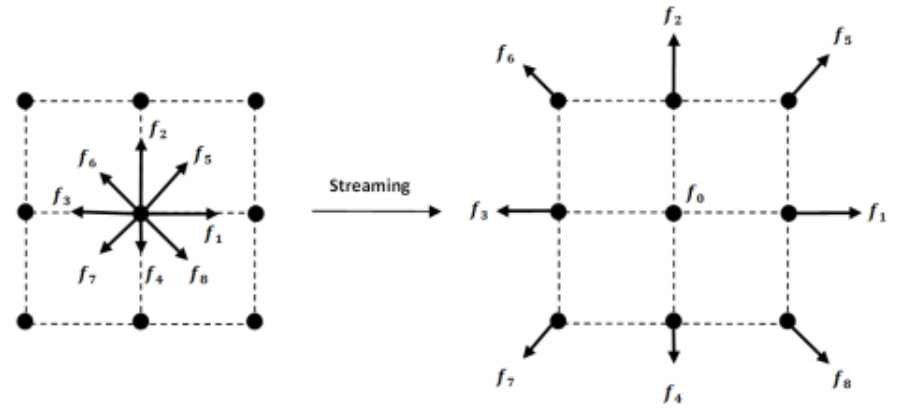
2.2.3. Ostale metode

Zadnjih godina osim navedenih javljaju se i novi pristupi. Najpoznatiji je boltzmanova latica (eng. *Lattice Boltzmann method"*). Za razliku od prethodnih, metoda ne rješava direktno Navier-Stokesove jednadžbe te se ne bavi makroskopskim svojstvima fluida. Fluid predstavlja skupom čestica te računa odnose među njima na puno nižoj razini - molekularnoj. U konačnici, ono što je poznatije kao makroskopsko svojstvo ustvari je samo rezultat ili manifestacija različitih procesa na nižoj razini. Čestice svojim sudarima izmjenjuju energiju i gibanje. Prostor se dijeli u ćelije te se za svaku definiraju dva svojstva – gustoća i brzina. Jedna iteracija algoritma odvija se u dva koraka:

- Kolizija
- Tok

U koraku kolizije za pojedinu ćeliju računa se gustoća i brzina. Ta brzina koristi se u koraku toka (Slika 2.2) za pomicanje čestica fluida u drugu ćeliju. Ovisno o strukturi latice moguće je ograničeno gibanje iz jedne ćelije u drugu. Postoje točno predefinirani smjerovi kojima se smije ići. Prednost ove metode je mogućnost

paralelizacije kod programske implementacije. Relativno je brza te puno bolje barata s rubnim uvjetima.



Slika 2.2 Prikaz jednog koraka toka (Preuzeto iz [13])

2.3. Simulacija i animacija fluida

Postoji specijalna grana računalne grafike zadužene za realističnu animaciju fluida. Iako ima velik broj dodirnih točaka sa CFD-om, ipak se razlikuje u nekoliko osnovnih stvari. Dok je kod CFD-a naglasak na realističnoj simulaciji toka i što realnijem ponašanju fluida pri zadanim uvjetima, kod animacije je naglasak na tome da nešto izgleda kao fluid. Nije nužno potrebno znanje o svim procesima i jednadžbama koje se događaju te će se puno manje kompleksni modeli pokazati zadovoljavajućima. Dvije glavne industrije koje koriste simulaciju fluida su filmska industrija i industrija razvoja igara.

Početci ove industrije najviše duguju Nicku Fosteru i Dimitrisu Metaxasu. Do njihovog preokreta fluid se na računalu simulirao bez računanja formula statike i dinamike fluida. Oni su 1996. uspjeli vizualizirati fluid u trodimenzionalnome prostoru na računalu koristeći Navier-Stokesove jednadžbe. Bio je to početak nove ere računalne grafike. Prvi koji je uspio napraviti simulaciju fluida koja je bila stabilna u svim uvjetima, a bila je relativno brza bio je Joe Stam. Počelo se najprije sa simulacijama relativno mirnih površina vode. Danas je simulacija fluida vizualno toliko dobro razvijena da je teško odrediti što je pravi fluid, a što računalno izračunato i generirano. Naravno, primat ima izgled. Njegova realnost postiže se

upotrebom taman dovoljno fizikalnih zakona da bi izgledalo realno, ali ne mora nužno biti sto posto točno. Također, izgled se postiže i nauštrb vremena kojeg u filmskoj industriji ima dovoljno.

3. Karakteristike OpenGL-a i osnovna struktura koda

3.1. OpenGL

„OpenGL grafički sustav je programsko sučelje grafičkom sklopovlju “ (Shrainer, Sellers, Kessenich, & Licea-Kane). Prva verzija razvijena je 1992. kao nastavak projekta IrisGL. Od tada je postao imperativ računalne grafike i dio koji se vrti „ispod haube“ u mnogobrojnim pogonima za grafičku simulaciju i izradu igara. U početku je bio dosta ograničen, na dosta stvari se nije moglo utjecati jer tadašnje grafičke kartice nisu bile programirljive kao što su danas. Kako se sklopovska oprema mijenjala, različitim ekstenzijama OpenGL moglo se pristupiti do tada programirljivim dijelovima hardvera. Međutim, od 2007 i OpenGL verzije 2.0 fiksni cjevovod je maknut te je OpenGL omogućio vlastito programiranje sklopovskih dijelova. Trenutno aktivna verzija na tržištu je 4.5, a puštena je 11. 8. 2014.

OpenGL ima klijent-server arhitekturu. Korisnik daje naredbe koje se zatim izvršavaju točno onim redoslijedom kojim su zadane. OpenGL barata samo s najjednostavnijim tipovima podataka – vrhovima (engl. verteksima), linijama i trokutima. Verteks je točka u homogenom 4D prostoru koja označava sjecište dviju linija ili rub jedne. Linija u matematičkom smislu odgovara onome što asociramo s dužinom kao najkraćom udaljenosti dviju točaka. Svaki složeniji objekt prikazuje se kao kompozicija ovih primitiva.

OpenGL ne ovisi o platformi i stoga ima ograničen skup mogućnosti kad je riječ o interakciji s korisnikom. Iz tog razloga razvijene su dodatne biblioteke. GLEW (OpenGL Extension Wrapper) nadopunjuje OpenGL s funkcionalnostima koje su specifične za pojedinu platformu. GLUT i GLFW imaju sličnu funkcionalnost. One nadoknađuju odsustvo sposobnosti za interakciju s korisnikom. Obrađuju ulazno–izlazne jedinice, inicijaliziraju kontekst, stvaraju prozor.

3.2. Sjenčari (engl. Shader)

„Laička definicija sjenčara bi bila program koji govori računalu kako da crta nešto na specifičan i unikatan način“. (JiJi) Postoji posebna kategorija jezika koji su rađeni za sjenčare. Osnovna podjela je na one koje se prevode prije ili za vrijeme izvođenja glavnog programa. U OpenGL-u jezik za shadere je GSLS (OpenGL Shading Language). To je visokoproceduralni jezik temeljen na C-u. Prevodi se za vrijeme izvođenja. Time se omogućuje interakcija s korisnikom. Postoji više vrsta sjenčara i svaki ima svoju funkciju u cjevovodu izrade prikaza. Njihove specifične funkcije usklađene su s funkcijama programirjlivog dijela GPU-a.

3.3. Cjevovod izrade prikaza

Slika 3.1 Cijevovod izrade prikaza

Cjevovod izrade prikaza (Slika 3.1) napravljen je tako da odgovara redoslijedu izvođenja GPU-a u svrhu postizanja maksimalnog ubrzanja. Glavna prednost ovakvog pristupa je slanje struja podataka. Čim se jedan podatak obradi u jednoj

fazi, automatski se šalje u drugu. Pritom ga u prethodnoj fazi zamjenjuje novi podatak.

3.3.1. Generiranje podataka

Sve podatke koji se šalju na obradu i iscrtavanje potrebno je učitati u posebnu strukturu podataka „buffer“ objekt (engl.). Pozivom funkcije „glGenBuffer()“ stvara se jedan takav objekt, ali tek poziv „glBindBuffer“ alocira memoriju i pridjeljuje stanje objektu.

3.3.2. Sjenčar vrhova (engl. *Vertex shader*)

„Vertex shader“ obavezna je faza koja kao ulaz prima jedan po jedan verteks – točku u homogenom 4D prostoru. Kao izlaz mora dati također jednu vrijednost, odnosno onoliko vrhova koliko primi toliko će ih i poslati dalje. Jedan sjenčar vrhova za svaki isti ulaz daje i identičan izlaz. Stoga, on ima svoju pričuvnu memoriju u koju privremeno pamti parove ulaza i izlaza tako da ako mu se ponovno pojavi neki ulaz za koji on već ima izračunatu vrijednost samo će je proslijediti dalje.

Zadatak sjenčara vrhova je pretvoriti koordinate objekta u koordinate svijeta. Što to zapravo znači? Zamislimo 3D točku u prostoru koja se nalazi na koordinati (3,0,1). Ako to želimo iscrtat na ekranu javlja se nekoliko problema Za početak, u odnosu na što se mjeri 3, 0 i 1? Gdje je ishodište koordinatnog sustava? U kojem se smjeru nalazi os x. Postaje jasno da je potrebno definirati neka pravila kojima će se jednoznačno odrediti iscrtavanje elementa na ekran. Definirat će se tri matrice čijim množenjem s koordinatama objekta dobivamo takvu transformaciju. Prva matrica je matrica modela. Svaki model ima svoj lokalni koordinatni sustav u kojem se nalazi. Njega je potrebno prebaciti u globalni koordinatni sustav svijeta u kojem se nalazi. Slijedeća je matrica pogleda. Ona definira način na koji se globalni koordinatni iscrtavan na sceni. Određena je trima vektorima

- Očište – gdje se nalazi promatrač
- Gledište – gdje se nalazi točka u koju je usmjeren pogled
- „View up“ vektor- gdje se nalazi gore

Projekcijska matrica definira način iscrtavanja. Postoje dva osnovna načina, a to su ortografski i perspektivni. Ako zamislimo snop paralelnih linija kod ortografske projekcije oni uvijek ostaju paralelni dok se kod projekcijske u daljini spajaju. Glavna mana ortografske projekcije je odsustvo dubine. Stoga realnom svijetu puno više odgovara projekcijska matrica. Ove matrice najbolje je ujediniti u jednu množenjem prije slanja sjenčaru radi jednostavnosti i smanjenja broja operacija. Redoslijed množenja je matrica modela, pogleda pa projekcije. Svaki vrh će se u sjenčaru pomnožiti s tom matricom.

3.3.3. Mozaik (engl. *Teselleation*)

U ovoj fazi dijelovi moguća je podjela ulaznih podataka na manje dijelove.??? Ono što se time može postići jest tzv. “Razina detalja” (eng. Level of Details). Što je promatrač bliže objektu, kompleksnost i razina detalja se povećava, vrijedi i obrnuto. To je analogno promatranju objekata u realnom svijetu gdje se razina detalja koje primjećujemo proporcionalno smanjuje s povećavanjem udaljenosti. Ovaj sjenčar nije obavezni dio cjevovoda. Odvija se u dvije proizvoljne faze koje su povezane jednom fiksnom fazom. Prva faza je TCS (engl. *Tesselation Control Shader*“). Tu se postavljaju parametri postupka. U srednjoj (fiksnoj) fazi se na temelju tih parametera primitiv dijeli na manje. U zadnjoj se fazi generiraju vrijednosti generiranih verteksa. Postoji posebna vrsta primitiva s kojima barata ovaj sjenčar, a to su tzv. zakrpe (engl. *patches*).

3.3.4. Geometrijski sjenčar

Geometrijski sjenčar nije obavezni dio cjevovoda. Kao ulaz prima jedan primitiv dok mu izlaz može biti nula ili više njih. Jedino je u ovoj fazi moguće generirati nove primitive ili odbaciti neke. Može se koristiti za veću razinu detalja, ali to mu nije primarna svrha te već postoji sjenčar s tim zadatkom. Geometrijski sjenčar najčešće se koristi za renderiranje u različitim slojevima.

3.3.5. Izrada prikaza

Renderiranje je proces kojim se 3D objekt preslikava na 2D površinu. Može se raditi na više načina. U OpenGL provodi se rasterizacijom. Rasterizacija obuhvaća dva koraka. U prvom se za 3D primitiv određuje gdje će ga se na ekranu točno smjestiti. Drugi korak pikselu na ekranu na kojem se iscrtava dio tog objekta pridružuje odgovarajuću boju.

3.3.6. Sjenčar fragmenata

Kao što je sjenčar vrhova baratao s vrhovima, tako sjenčar fragmenata barata s fragmentima. Fragment je konačni produkt rasterizacije. Sjenčar na ulaz prima jedan fragment te mu pridružuje boju i dubinu. U ovoj se fazi radi i preslikavanje teksture.

3.3.7. Iscrtavanje scene

Za iscrtavanje scene može se koristit jednostruki ili dvostruki spremnik. Kad se koristi jednostruki spremnik onda se i iscrtavanje scene i prikaz odvija u jednom spremniku. Može doći do usporenog osvježavanja ekrana ili kombinacije slike kojoj dio pripada prethodnoj fazi iscrtavanja, a dio slijedećoj. Puno bolja alternativa je dvostruki spremnik. U jedan spremnik se iscrtava scena, a drugi se koristi za prikaz. Nakon toga funkcionalnosti ta dva spremnika se mijenjaju. U drugom će se iscrtavati scena dok će se prvi koristiti za prikaz.

3.4. Struktura programa

Struktura glavnoga dijela programa svih navedenih implementaciji bit će ista te će se stoga opisati samo jednom.

Pseudokod:

```
Inicijaliziraj_GLFW()

Inicijaliziraj glew()

Kreiraj_Porozor()
```

```
Generiraj_potrebne_objekte();
dok (ne_treba_zatvorit_prozor)
        ako je potrebno
        generiraj nove objekte()
        obnovi_izracune()
        prikaziScenu()
oslobodi memoriju
```

Na početku programa potrebno je izvršiti potrebne inicijalizacije i stvoriti prozor. Nakon toga generiraju se početni objekti i uvjeti. Program ulazi u petlju koja se izvršava sve dok korisnik ne zatvori prozor. U jednom koraku petlje obavljaju se potrebni izračuni fizikalne simulacije te se ponovno iscrtava scena.

Za sve trodimenzionalne prikaze potrebno je bilo napraviti odgovarajuće sjenčare kako bi se objekt mogao smjestiti na scenu. U sklopu ovog rada napravljeni su samo sjenčar vrhova i sjenčar fragmenta. Napisana je posebna klasa koja barata s njima. U toj klasi učitavaju se sjenčari, kompajliraju i povezuju sa programom. U glavnom programu definiraju se vektori modela, pogleda i vektor koji označava gdje se na sceni nalazi gore. Korisničkim tipkama se može upravljati pogledom, stoga je nužno definirati ih u glavnom programu. Na temelju njih pravi se matrica pogleda. Matrica modela objektu samo dodaje 4 koordinatu. Za projekciju odabrana je perspektivna matrice jer daje realniju sliku. Matrice su pomnožene u glavnom programu te se daju na ulaz sjenčara vrhova. U njemu se svaki verteks množi sa tom matricom. Zadatak sjenčara fragmenata je pridruživanje boje. Česticama se pridružuje plava boja jer se simulira voda. Jedan fragment dobiva 4 vrijednosti boja (RGBA). Prva tri označuju crvenu, zelenu i plavu komponentu. Slovo „A“ označuje alfa kanal i koristi se da za davanje prozirnosti.

Svi programi pisani su u C++ isključivo zbog brzine izvođenja. C++ je daleko najbrži objektno orijentirani jezik. Kao razvojno okruženje korišten je „Microsoft Visual Studio 2015“. On u sebi ima mogućnost ugrade paketa „NupenGL“ koji dolazi sa slijedećim bibliotekama:

- Freeglut
- Glew
- Glfw

Od navedenih korištene su glew i glfw. Za lakši rad s matricama i vektorima korištena je dodatna biblioteka *glm* (OpenGL Mathematics), matematička biblioteka pisana za C++ i OpenGl. Korištena verzija OpenGL-a je 3.1. To je posljednja verzija koju podržava Intelova grafička kartica na kojoj se se vrtjele simulacije.

4. Simulacija gibanja jednostavnim sustavom čestica

4.1. Ideja programa

Slika 4.1 Fontana (Slikala Samanta Kretić)

Pogledajmo primjer vodoskoka iz kojega voda teče dijagonalno prema gore (Slika 4.1). Izgledom podsjeća na ono što se u fizici naziva kosi hitac – složeno gibanje koje se sastoji od jednoliko ubrzanog i jednolikog gibanja po pravcu u različitim smjerovima. Što bi se dogodilo ako bi se voda zamijenila pijeskom ili nekim drugim skupom krute tvari malog volumena i velike brojnosti. Primjećujemo da bi gibanje ostalo vrlo slično. Ukoliko onda želimo simulirati neko takvo gibanje je li zbilja nužno računati površinsku napetost, tlak vode i viskoznost ili je pak moguće primijeniti pravila koja bi primijenili i na kosi hitac? Odgovor je prvo, nije nužno računati. Moguće je postići zadovoljavajuću aproksimativnu simulaciju tako da na ekranu

prikažemo malene kapljice vode koje se gibaju pod utjecajem gravitacijske sile. Ne mora tu nužno biti riječ o kosom hitcu, može se primijeniti bilo kakvo složeno gibanje ovisno zadatku simulacije. Prva i najjednostavnija implementacija napravljena u okviru diplomskog rada je simulacija jednog takvog vodoskoka. Fluid će biti dekomponiran na malene čestice i na njega će se primijeniti formule iz dinamike krutih tijela.

4.1.1. Sustav čestica u kontekstu računalne grafike

Postoji skup objekata koji za razliku od tipičnih krutih tijela nema jasno definiranu površinu, volumen ili točno zadano karakteristično ponašanje. U takve objekte mogli bi ubrojiti padaline, vatromet, eksploziju. Sama za sebe instanca takvog objekta nema veliko značenje, tek u zajednici sa ostalima dobiva smisao i cijela kompozicija nešto predstavlja. William Reeves je 1983. godine objavio rad „Sustav čestica – Tehnika za modeliranje klase nejasnih objekata“. Uveo je pojam čestice kao jedne instance takvog objekta i sustava čestica kao spomenute kompozicije. Iako je navedeni rad prvi sustavni opis jednog takvog sustava, on se bez imena već prije pojavljivao u industriji igara i filmova. Rani sustavi čestica nisu podrazumijevali međusobnu interakciju čestica, ali današnji se koriste i za to. Prema Reeveseu[1] objekti koje ćemo predstavit česticama za njega imaju nekoliko osnovnih svojstava. Nisu predstavljeni skupom poligona te je njihova putanja često poprilično složena. Čestice imaju sposobnost tzv. „rađanja i umiranja“ odnosno kreiranja i uništavanja. To sustav čestica čini dinamičkim sustavom kojem se mijenja broj članova. Razdoblje u kojem postoji jedan objekt čestice naziva se njezinim životnim ciklusom . Sastoji se od triju osnovnih faza

- Generiranje čestice
- Život
- Umiranje čestice

Generiranje čestice i njezin život temelji se na stohastičkim procesima. Dio svojstava odabire se slučajnim generiranjem brojeva što uvelike olakšava posao

[1] Za više detalja o navedenoj tematici vidi (Reeves, 1983)

programerima. Generiranje čestica može se obaviti jednom na početku ili više puta tijekom izvođenja algoritma. Kod generiranja čestici daju se njezina osnovna svojstva, a to su:

- Početni položaj
- Iznos i vektor smjera brzine
- Vizualna svojstva kao što su boja, veličina i oblik
- Životni vijek

Čestica tijekom svojega života mijenja položaj u ovisnosti u o vektoru brzine. Prilikom svake iteracije algoritma život joj se smanjuje za jedan. Umrle čestice brišu se iz memorije. Kada bi se konstantno generirale nove čestice bez prethodnog brisanja starih, očigledno je da bismo mogli ostati bez resursa u memoriji. Dobro je stoga definirati neke dodatne uvjete u kojim nam čestica više neće biti zanimljiva. Jedan od takvih može biti da se makla s vidljivog dijela scene ili joj se veličina toliko smanjila da je više uopće ne vidimo. U takvim slučajevima bespotrebno je čekati da joj se preostali životni vijek smanji na nulu da bismo je pobrisali.

Nužno je odrediti način na koji će se čestice iscrtavati na ekran. Najjednostavniji pristup je iscrtavanje osnovnog primitiva - verteksa (točke). Može se prikazati i poligonom s teksturom. Tada je potrebno prilagoditi rotaciju poligona tako da je normala ravnine usmjerena prema promatraču. Takvom poligonu moguće je pridružiti i teksturu.

4.2. Struktura čestice i koda

Osnovni pseudokod algoritma odgovara prethodno spomenutom. Ono što se posebno ističe je struktura čestice koja predstavlja maleni dio volumena fluida. Modelirana je zasebnom klasom i ima sljedeća svojstva koja odgovaraju osnovnim svojstvima čestice navedenim u prethodnom potpoglavlju:

- Pozicija u prostoru (r)
- Brzina (v)

- Veličina
- Životni vijek
- Varijabla koja govori živi li čestica vječno

Pozicija i brzina predstavljene su 3D vektorom. Na početku programa inicijalizira se početna vrijednost pozicije i brzine svake čestice. U svakom koraku glavne petlje generiraju se nove čestice te se zatim računaju nove vrijednosti starih. Nova vrijednost vektora pozicije i brzine za svaku pojedinu česticu računa se na sljedeći način

$$\vec{v}(\mathrm{t} + \Delta t) = \vec{v}(\mathrm{t}) + \vec{a}$$

$$\vec{r}(\mathrm{t} + \Delta t) = \vec{r}(\mathrm{t}) + \vec{v}\,(\mathrm{t} + \Delta t) * \Delta t;$$

Akceleracija će isključivo biti pod utjecajem sile teže koja djeluje negativno od smjera osi y. U svakom vremenskom koraku Δt vrijednosti brzina promijenit će se na način koji je prikazan slijedećim formulama.

$$v_x = v_x - a_x * \Delta t = v_x + 0 = v_x$$

$$v_y = v_y - a_y * \Delta t = v_y - g$$

$$v_z = v_z - a_z * \Delta t = v_z + 0 = v_z$$

Nakon što su izračunate nove pozicije, one se spremaju u polje koje će se poslati spremniku za iscrtavanje. Nakon što se sve nacrta, sve čestice koje su pale ispod 0 po os y brišu se. Stoga, u početku broj čestica raste, no nakon nekog vremena se stabilizira. Otprilike jednak broj čestica se generira i izbriše u jednom vremenskom koraku Δt.

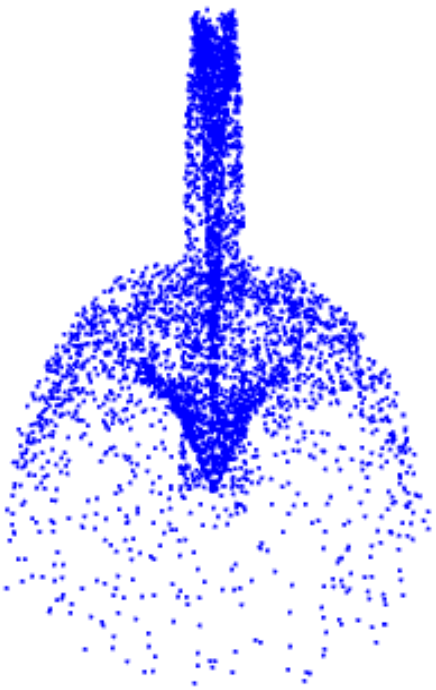
4.3. Parametri simulacije

Ovaj algoritam upotrijebiti će se za simulaciju fontane u dva sloja pri čemu jedan ide više u visinu, dok se drugi pruža više u širinu. Postoji nekoliko osnovnih kontrolnih parametra

- Učestalost generiranja čestica

- Broj novonastalih čestica po jednom generiranju
- Veličina vremenskog koraka

Određivanje ovih parametara bilo je eksperimentalno. Ne postoji računski način za odrediti sve parametre jer dosta toga ovisi i o jačini grafičke kartice i radnoj memoriji kompjutora. Kao dobar vremenski razmak pokazao se 0.01. Dovoljno je malen da simulacija ne eksplodira, a opet ne previše da bi znatno usporio. Svako 4 koraka glavne petlje generira se nova populacija veličine 80, od čega 2/3 idu u donji, a 1/3 u gornji sloj fontane. Sve čestice počinju gibanje u ishodištu O(0,0,0), ali svaka dobiva vektor brzine u različitom smjeru. Orijentacija u odnosu na osi x i z određuje se stohastički. Apsolutna vrijednost vektora brzine različitih čestica istog sloja relativno je slična uz male varijance kako bi se izbjegao artificijelan izgled simulacije. Nije određena apsolutna vrijednost životnoga vijeka jedne čestice. Česticu ćemo proglasiti „mrtvom“ ukoliko nam više nije vidljiva, a to se događa kad vrijednost y komponente padne ispod 0. Smatra se da je tada ispod razine vode u fontani te je ne možemo uočiti. Na slici 4.2 može se vidjeti konačan produkt implementacije.



Slika 4.2 Prikaz fontane

4.4. Prednosti i nedostatci ovakvog pristupa

Osnovna prednost ovakvog promatranja jest brzina kojom se odvijaju izračuni, pa samim time i cijela simulacija. Kod izračuna vrijednosti jedne čestice potrebni su samo podaci pohranjeni u toj čestici. Složenost jednog prolaska petlje jest O(n) pri čemu je n broj čestica. Kod jednostavnijih simulacija ovakav naivan pristup često je zadovoljavajući. Međutim, ukoliko želimo prikazati površinu fluida koja se miče, uviđamo kako ovaj pristup nikako nije dobar.

5. Aproksimacijska visinska mapa

Simulacija česticama nije dobar odabir ukoliko se želi pokazati samo površina nekog fluida (ocean, rijeka, jezero). Bespotrebno kompliciranje računski može biti zahtjevno, a u grafičkom smislu neće pretjerano uljepšati simulaciju. „Ako smo zainteresirani za animaciju dvodimenzionalne plohe, bilo bi pretjerana simulacija cijelog tijela vode.“ (Bridson & Muller-Fischer, 2007) U tom slučaju može se koristiti algoritam aproksimacije visinskom mapom. Jednostavan je za implementirati i postiže vizualno atraktivne rezultate.

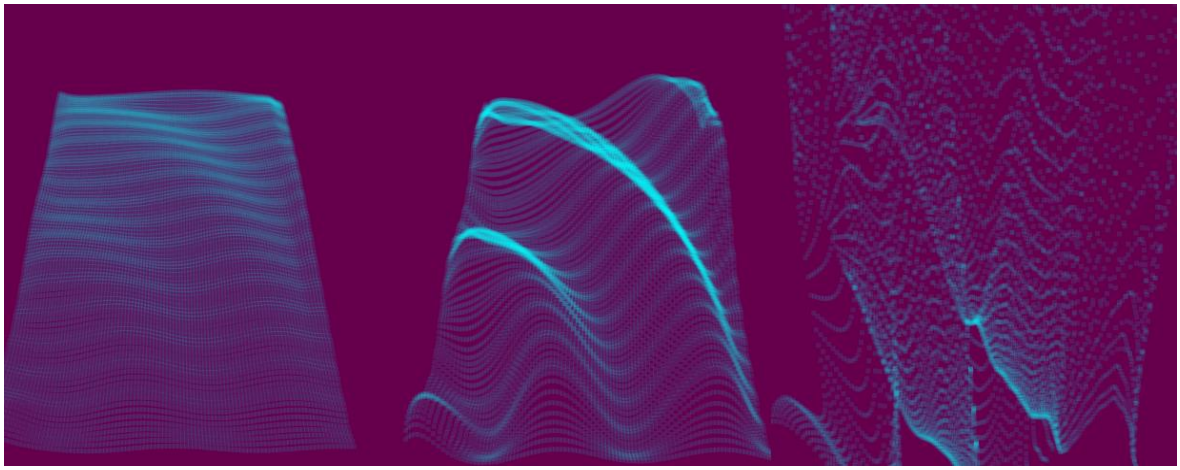
Površina koja se simulira podijeli se na 2D mrežu. Što je mreža gušća, to će vizualizacija biti detaljnija. Za svaki element te mreže pamte se slijedeći podaci – visina (u) tog stupca i brzina kojom se visina u stupcu mijenja (v). Za početnu vrijednost može se postaviti oblik mreže kakav god se želi. Pseudokod za mijenjanje izgleda mreže preuzet je iz SIGGRAPH-ovog tečaja za simulaciju fluida. (Bridson & Muller-Fischer, 2007)

Pseudokod:

```
za svaki i,j
    v[i,j] = (u[i-1,j] + u[i+1,j] + u [i,j-1] + u[i,j+1]) / 4 – u[i,j]
    v[i,j] * = tezinski_faktor_manji_od_jedan
    u[i,j]+=v[i,j]
```

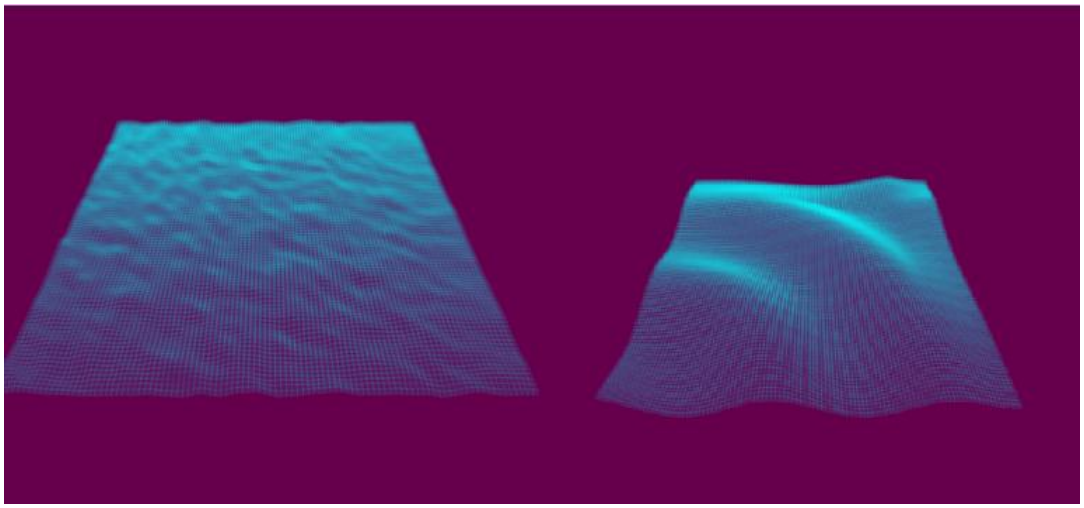
U literaturi se kao težinski faktor za smanjenje brzine predlaže 0.99. Kod implementacije algoritma taj se broj pokazao prevelik i s njime bi nakon nekoliko

iteracija simulacija eksplodirala (Slika 5.1). Idealnim se pokazao faktor koji iznosi 0.68.



Slika 5.1 Eksplozija simulacije za prevelik parametar prigušenja brzine

Treba definirati što se događa kod rubnih uvjeta, točnije kada su i i k manji od nula ili veći od maksimalne dužine (širine). Najjednostavnije je tada ih prilagoditi vrijednosti koja je do njih tako da u[-1,j] = u[0,j] i analogno tome za sva 4 ruba. Konačan izgled simulacije može se vidjeti na slici 5.2.

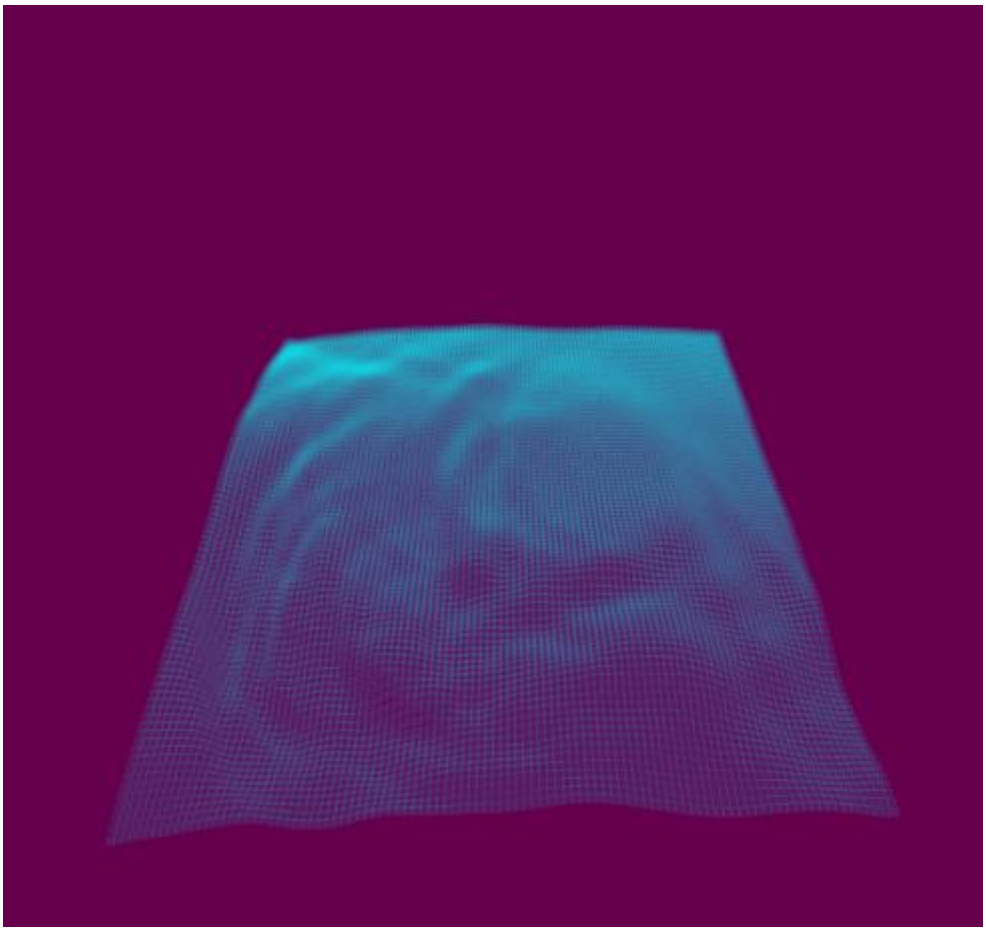


Slika 5.2 Simulacija za različite početne funkcije

5.1. Implementacija

Cilj programa je simulirati vodenu površinu koja može počinjati iz nekoliko različitih položaja. Može biti skroz mirna za početak ili uhvaćena u gibanju. Program će omogućiti interakciju s korisnikom na način da korisnik može tijekom izvođenja birat početni izgled ili povlačit miš po vodi i time uzburkavati površinu. Na početku se definiraju nužni parametri za izvođenje simulacije, a to su širina i duljina plohe te granulacija mreže. Visinu u i brzinu v se pohranjuje u jednodimenzionalnu strukturu podataka vektor. Ako se želi pristupiti elementu koji se nalazi na mjestu i i j, indeks će biti i*broj_celija_duljine + j.

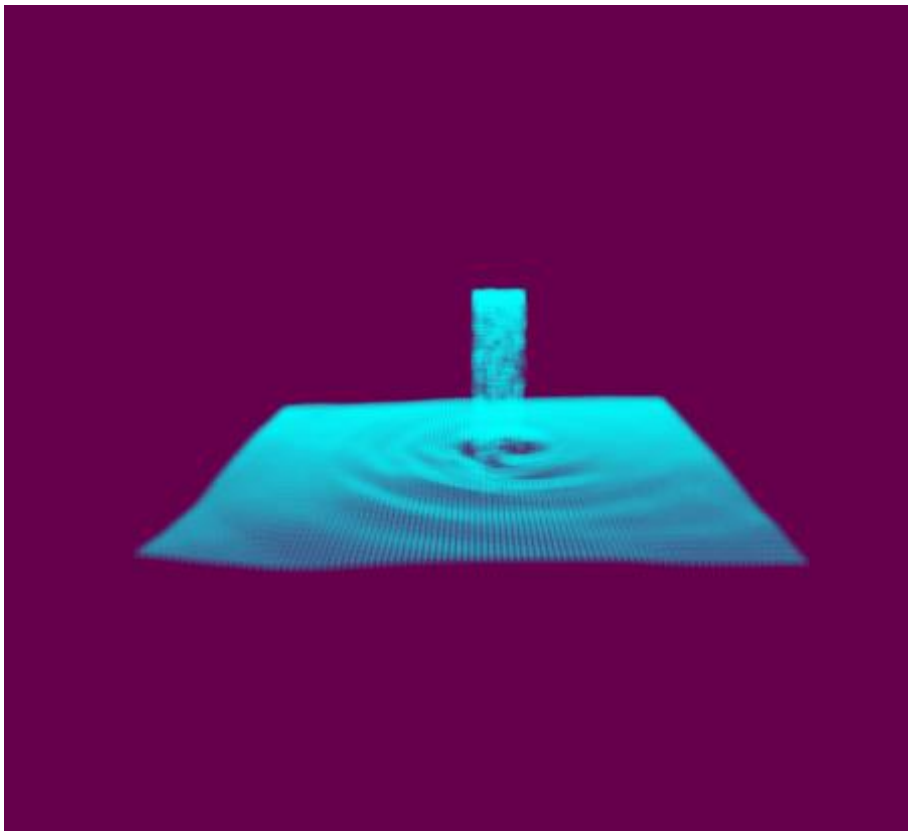
Korisnik na simulaciju može utjecati preko miša i tipkovnice. To je ostvareno pomoću GLFW standardnih funkcija za baratanje s takvim događajima. Definiraju se funkcije koje se pozivaju kada korisnik stisne tipku na tipkovnici ili mišu. Ukoliko korisnik želi ponovno pokrenuti simulaciju dovoljno je kliknuti na tipku S. Kako bi se omogućilo promatranje različitih početnih uvjeta, odabirom broja na tipkovnici bira se funkcija koja će postaviti inicijalnu visinu cijele mreže. Tipke gore, dolje, lijevo i desno služe za pomicanje očišta. Gledište je fiksno i nalazi se u ishodištu 3D koordinatnog sustava. Osim toga, ostvarena je još jedna funkcionalnost – korisnik povlačenjem miša po ekranu može uzburkati površinu. Od trenutka kada korisnik spusti lijevu tipku miša pa do trenutka kad je otpusti, sve točke na ekranu koje obiđe pamte se u za to namijenjeno polje. Nakon toga provodi se transformacija koordinata ekrana u koordinate svijeta. Rubovi ekrana odgovaraju rubovima površina, a točke u sredini računaju se tako da omjer ostaje isti. Primjerice, klikne li se na točku koja na ekranu odgovara 90% širine i 10% visine, i na simulaciji će se odabrati točku koja odgovara tom omjeru. Problem je što se u slučaju kada korisnik prebrzo miče miš po ekranu ne uspiju uhvatiti sve točke. Zato se između svakoga para točaka povlači imaginaran pravac te se sve ćelije koje leže na tom pravcu i nalaze se između zadanih točaka također uključuju. Svim takvim točkama mijenja se visina što se onda dalje propagira kroz simulaciju (Slika 5.2).



Slika 5.3 Valovi uzrokovani interakcijom s korisnikom

5.2. Dodavanje čestica i njihova kolizija

Kako bi se simulacija učinila zanimljivijom, dodane su i čestice koje bi trebale simulirati vodu koja pada na površinu. Čestice su napravljene na način koji je opisan u prethodnim poglavljima. Jedino je dodatno potrebno omogućiti provjeru dolaska čestice do površine vode. Svaki put kada se čestici obnovi pozicija, potrebno je na mreži naći koji indeks odgovara poziciji čestice koju obnavljamo. Zatim se provjerava visina vode na tom indeksu. Ukoliko se čestica nalazi ispod vode, briše se. Zatim se simulira sila kojom je ta čestica djelovala na sličan način kao i kad je korisnik kliknuo miš (Slika 5.3).



Slika 5.4 Interakcija mape sa sustavom čestica

5.3. Prednosti i mane visinskih mapa

Glavna prednost uporabe visinskih mapa je brzina izračuna. Svaka ćelija za svoju vrijednost koristi samo 4 neposredna susjeda. Opterećenje memorije također nije veliko jer se pamte podaci samo površine, za razliku od eulerovog ili lagrangeovog pristupa. Mana ovog pristupa je ograničenost izbora toka fluida. Mogu se simulirati relativno mirni uvjeti jer se u svakom trenutku na jednoj ćeliji može nalaziti samo jedna razina vode. Rješenje tog problema bilo bi dodati sustav čestica na kritičnim točkama gdje pretpostavljamo da bi se fluid trebao burnije ponašati. Ovo pretpostavlja određeno apriorno znanje o problemu koji simuliramo i gdje bi se fluid mogao ponašati na specifičan način.

6. Metoda zaglađene hidrodinamike čestice (SPH)

6.1. Osnove SPH

Kada je 1977. godine izumljena metoda zaglađene hidrodinamike čestice (SPH), nije bila namijenjene CFD. Monaghan i Gingold te Lucy metodu su razvili sa svrhom rješavanja astrofizičkih problema. Međutim, metoda se pokazala puno bržom od Eulerove i pogodnijom za simulaciju turbulentnijih tokova. Nije dugo trebalo da tehnika pronađe svoj put do CFD-a.

U središtu metode nalazi se čestica kao predstavnik malog dijela volumena fluida te nositelj makroskopskih svojstava kao što su brzina, gustoća, tlak. Ova tvrdnja korijene vuče iz ranije spomenute teorije kontinuuma koja osigurava postojanost makroskopskih svojstava i na ovako malom redu veličine. SPH kaže da vrijednost svojstva pojedine čestice možemo dobiti zbrajanjem svojstava susjednih čestica pomnoženih s odgovarajućom zagađujućom funkcijom (kernelom). Matematička podloga te tvrdnje bit će objašnjenja i dokazan u nastavku. Neka je dana funkcija f(x). Prema definiciji, jedno od osnovnih svojstava delta (dirac) funkcije $\delta(x)$ jest:

$$\int_{-\infty}^{+\varphi} f(x) * \delta\,(x-a)\,dx = f(a)$$

Prebacit ćemo to u 3D domenu i funkciju f(x) zamijeniti s nekim svojstvom A(r) nad volumenom V. Imamo sljedeće

$$\int_{V} A(r') * \delta(r-r')\,dr' = A\,(r)$$

Pri čemu su r i r' vrijednosti unutar volumena V. Dirac delta funkcija zamjenjuje se s kernel funkcijom W koja djeluje na radiju su h.

$$\int_{V} A(r') * W(r-r',h)\,dr' = A\,(r)$$

Prema (Nataesescu & Gavrila, 2011) kernel je funkcija sa slijedećim svojstvima:

$$\int_V w(r - r')\,dx = 1$$

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

Kernel je, također simetrična funkcija koja je uvijek veća ili jednaka 0. Kada ide iz negativne beskonačnosti u 0 mora biti monotono rastuća, dok je od 0 do pozitivne beskonačnosti monotono padajuća. Najveću vrijednost poprima u 0 odnosno kada je r=r'.

Ranije spomenuto svojstvo A(r) sada će se zapisati u diskretnoj domeni za svaku pojedinu česticu i. Pri tom se u obzir uzima činjenica da je $m = \Delta V \cdot D$

$$A_i = \sum_j \frac{m_j}{\rho_j} A(r_j) * W(r_i - r_j, h)$$

Ako se želi izračunati neko svojstvo A čestice i, potrebno je zbrojiti to isto svojstvo svih susjednih čestica zaglađeno odgovarajućom funkcijom.

6.2. Osnovni algoritam

Algoritam koji je implementiran temelji se na radu „Smoothed Particle Hydrodynamics on GPUs" autora Takaira Harade, Seiichi Koshizukaa i Yoichira Kawaguchija. (Harada, Koshizuka, & Kawaguchi). Navier- Stokesova jednadžba u Lagrangeovom sustavu daje nam akceleraciju pojedine čestice.

$$\frac{d\vec{u}}{dt} = g - \frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2 u$$

Algoritam će se provoditi u tri koraka. U prvom će se izračunati sve komponente i ukupna akceleracija. Zatim će se vektor brzine promijeniti za iznos akceleracije. U konačnici će se izračunati nove pozicije čestica tako da nova brzina nadoda vektoru pozicije. Akceleracija će se mijenjati zbog triju komponenti. Gravitacija će djelovati u smjeru sile teže, tlak će gurati u smjeru negativnog gradijenta s mjesta višega na mjesto nižega tlaka. Viskoznost će djelovati u smjeru akceleracije.

6.2.1. Gustoća čestice

Kako bi se izračunala gustoća čestice, svojstvo A(r) zamijeniti će se sa D.

$$\rho_i = \sum_j \frac{m_j}{\rho_j} \rho_j * W(r - r_j, h) = \sum_j m_j * W(r_i - r_j, h)$$

6.2.2. Tlak

Kad se izračunala gustoća svake čestice, može se krenuti na izračun tlaka. Ovdje će se koristiti svojstvo linearnosti derivacije pa se

$$A_i = \sum_j \frac{m_j}{\rho_j} A(r_j) * W(r_i - r_j, h)$$

može pisati kao

$$\nabla A_i = \sum_j \frac{m_j}{\rho_j} A(r_j) * \nabla W(r_i - r_j, h)$$

Računa se $\frac{\nabla p}{\rho_i}$

$$\frac{\nabla p_i}{\rho_i} = \nabla \left(\frac{p_i}{\rho_i}\right) + \frac{P_i}{\rho_i^2} \nabla \rho = \sum_j \frac{m_j}{\rho_j} * \frac{P_j}{\rho_j} * \nabla W(r - r_j, h) + \sum_j m_j * \frac{P_i}{\rho_j^2} * \nabla W(r_i - r_j, h)$$

$$\frac{\nabla p_i}{\rho_i} = \sum_j \frac{m_j}{\rho_j} * \left(\frac{P_j}{\rho_i^2} + \frac{P_i}{\rho_j^2}\right) \nabla W(r_i - r_j, h)$$

U pojedinoj čestici tlak p_i računat će se kao k * $((D - D_0)$. K je konstanta plinova, D je trenutna gustoća, a D_0 je gustoća pri mirovanju.

6.2.3. Viskoznost

Zadnje se računa akceleracija zbog viskoznosti koja iznosi $\frac{\mu}{\rho_i} \nabla^2 u$. I ovdje će se iskoristi svojstvo linearne derivacije koje kaže da je

$$\nabla^2 A_i = \sum_j \frac{m_j}{\rho_j} A(r_j) * \nabla^2 W(r_i - r_j, h)$$

Slijedi:

$$\frac{\mu}{\rho_i} \nabla^2 u = \sum_j \frac{m_j}{\rho_j} (u_j - u_i) * \nabla^2 W(r_i - r_j, h)$$

6.3. Odabir kernel funkcije

Za kernel funkciju odabrana je funkcija (Harada, Koshizuka, & Kawaguchi)

$$W(r_i - r_j, h) = \frac{315}{64\pi h^9} * \left(h^2 - \|r_i - r_j\|^2\right)^3$$

$$\nabla W(r_i - r_j, h) = \frac{45}{\pi h^6} * (h - \|r_i - r_j\|)^2 * \frac{r_i - r_j}{\|r_i - r_j\|}$$

$$\nabla^2 W(r_i - r_j, h) = \frac{45}{\pi h^6} * (h - \|r_i - r_j\|)$$

Pseudokod algoritma je slijedeći:

Za svaku cesticu:

Izracunaj gustocu

Za svaku cesticu:

Izracunaj akceleraciju zbog promjene tlak

Izracunaj akcelraciju viskoznosti

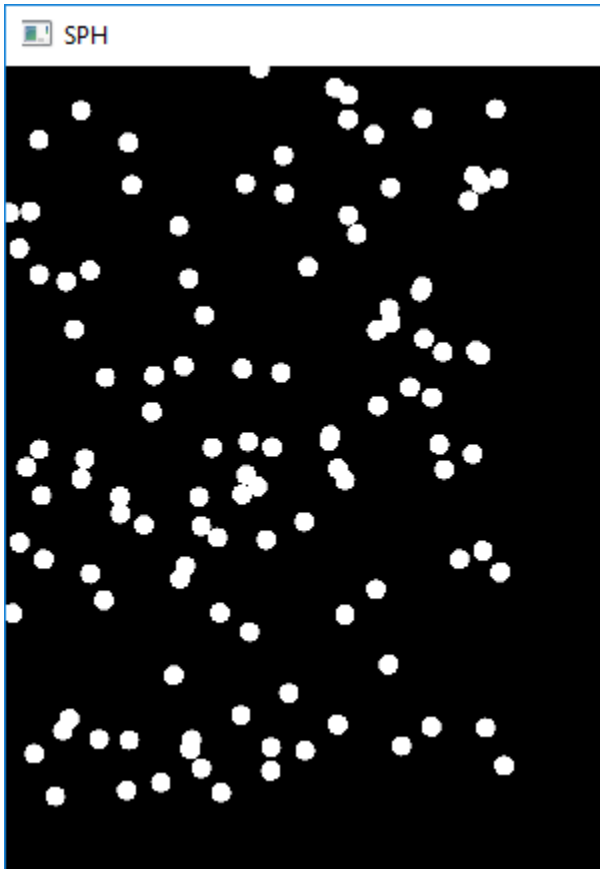
Izracunaj ukupnu promjenu brzine

Povecaj brzinu

Promjeni polozaj cestice

6.4. Implementacija rubnih uvjeta

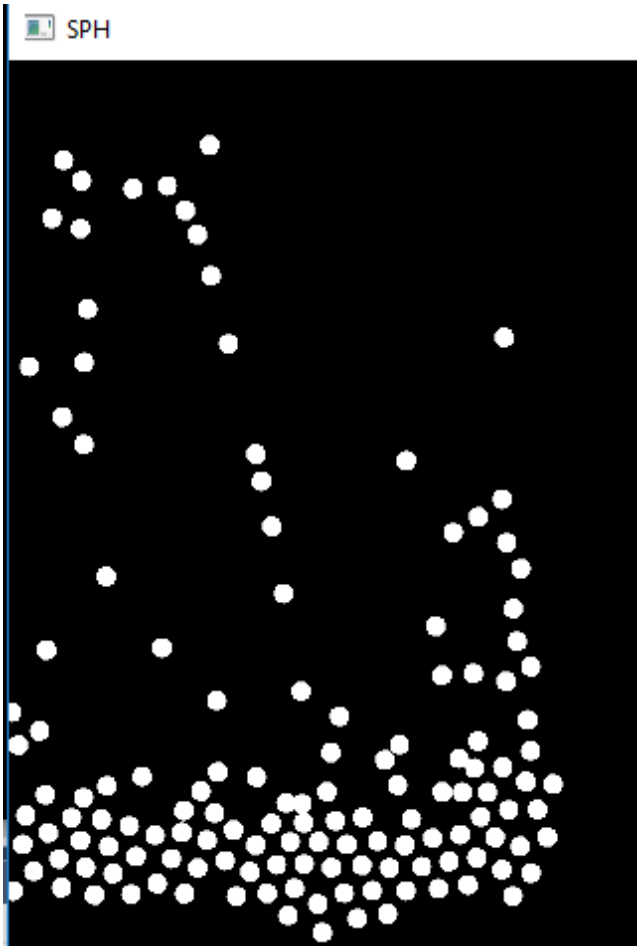
Potrebno je definirati način na koji se fluid ponaša kad se susretne s podlogom. Odbija li se ili pak ostaje mirovati? Za početak, iz dinamike je preuzeta ideja savršenog elastičnog sudara. Pretpostavka je bila da bi trebalo funkcionirati dobro s obzirom da taj zakon jamči očuvanje količine gibanja i kinetičke energije. Implementacija je bila sljedeće – kod svake kolizije čestice s rubom promjeni se smjer parametra brzine okomitog na plohu sudara. Odmah pri prvom pokretanju bilo je jasno da ideja nije bila dobra. U realnim uvjetima fluidi se nikad ne odbijaju savršeno elastično, dio energije gubi se pri sudaru. Simulacija je poslije svakog sudara bivala sve kaotičnija i kaotičnija dok u konačnici fluid nije izgubio oblik i čestice su se kaotično gibalo po ekranu.



Slika 6.1 Kaotično gibanje po ekranu

Nakon tog neuspjeha trebalo je pronaći fizički realnije rješenje. Kada bi se u različite slojeve vode koja teče dodao obojani fluid, mogla bi se promatrati brzina toka u različitim slojevima. Tada bi se primijetila zanimljiva činjenica, a to je da se dijelovi

fluida tik uz podlogu koja miruje ne kreću. Ta se pojava naziva uvjetom bez klizanja (engl. „No-slip condition“). Uzrok tome je viskoznost fluida. To je ujedno i sljedeći pokušaj implementacije sudara čestice sa rubom. Sada svaki put kada neka čestica dođe do ruba, brzina joj se postavlja na nul vektor. Simulacija se pokazala zadovoljavajućom. Jedini problem je što ponekad pojedina čestica takne rub i ukoliko oko nje nema nekog susjeda da djeluje na nju, može zaglavit zalijepljena tamo.



Slika 6.2 Lijepljenje čestica za rubove

To se pokušalo riješiti tako da se pri sudaru s čvrstom podlogom na nulu postavi samo ona komponenta brzine koja je okomita na udarenu plohu. To je dovelo do drugog problema. Čestice su imale sklonost šetati po rubu lijevo i desno od točke sudara, ali se izbjeglo lijepljenje čestica s rubova.

6.5. Ubrzavanje algoritma

Asimptotska složenost opisanog algoritma je $O(n^2)$. Povećavajući broj čestica, čija je veličina nužan uvjet što realnije simulaciji, brzina simulacije znatno otpada. No, moguće je ostvariti znatno ubrzanje uvođenjem susjedstva. Kada bi svaka čestica znala svoje susjede, ne bi morala pretraživati čitav prostor rješenja već samo one koje utječu na nju. Kontrolni parametar h govori granicu kada čestice prestaju djelovati jedna na drugu. Kada bi se čitav prostor simulacije podijelio na ćelije veličine parametra h, svaka čestica znala bi da joj se susjedi mogu nalaziti ili u vlastitoj ćeliji ili u ćelijama koje su joj susjedne. Ako je broj tih ćelija N znači da bi se u svakoj u prosjeku nalazilo $\frac{N}{b}$ čestica. Iz tog proizlazi da bi prosječan broj susjeda jedne ćelije bio $8 * \frac{N}{b}$. Označimo taj broj sa *a*. Sada bi složenost jednog prolaska algoritma bila O(aN).

Ubrzanje će biti na štetu memorijskog prostora. Na početku programa potrebno je inicijalizirati rešetku koja će se pri svakom prolasku glavne petlje puniti indeksima čestica koji se u prostoru nalaze na mjestu te ćelije. Poslije, kod traženja susjeda pojedine čestice, pogledat će se vrijednosti zapisane u toj ćeliji, i po jednoj manje (više) u svim smjerovima.

6.6. Problemi i utjecaj parametara

Glavni problem pri implementaciji SPH bio je namještanje parametara. Navier-Stokesove jednadžbe iznimno su osjetljive na parametre. Čak i promjena od 0.01 može označavati ključnu razliku između uspješne simulacije i neuspješne. Parametri koji su zadavali najviše problema bili su D_0 i k. Prvi je gustoća fluida pri mirovanju, a drugi konstanta. Oni direktno utječe na tlak koji je u simulaciji definiran kao k($D - D_0$). Ukoliko tlak kojim čestice djeluju jedna na drugu bude prevelik, simulacija se raspadne po ekranu i eksplodira. S druge strane, ukoliko je premalen, čestice plove jedna prema drugoj i cijela se simulacija uruši sama u sebe.

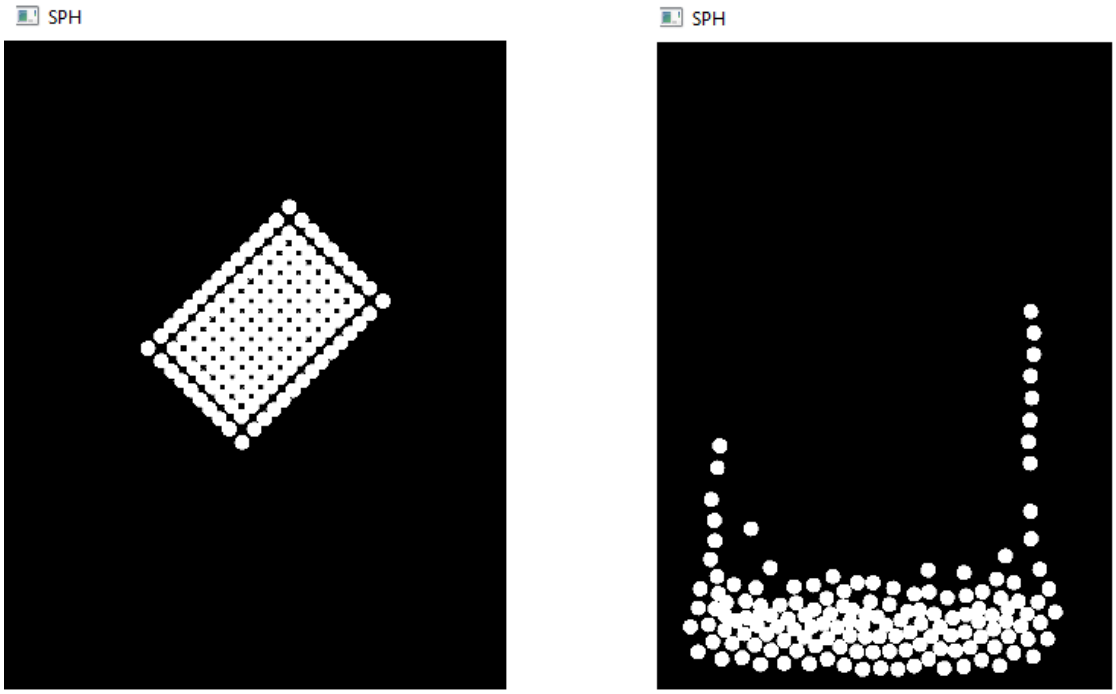
Drugi problematičan parametar bio je određivanje mase pojedine čestice. Ukoliko se postavi premalena masa, čestice se kaotično gibaju po ekranu te se simulacija više može primijeniti na ponašanje plinova nego na ponašanje tekućina.

Parametar h određuje područje koje se zaglađuje kernel funkcijom. Premalen parametar h uzrokuje to da niti jedna čestica ne djeluje jedna na drugu i cijela se simulacija uruši u par točaka na ekranu. Prevelik h znatno povećava vrijeme računanja jer više čestica smatramo susjednim te za svaku moramo obaviti potrebne proračune gustoće, tlaka i viskoznosti.

Parametar vremenskog koraka određuje diskretizaciju vremenske domene. On mora biti dovoljno velik da simulaciji ne bi bila previše usporena. Također mora biti dovoljno malen kako bi osigurao realno ponašanje simulacije. U obzir se treba uzeti činjenice da se sve diferencijalne jednadžbe aproksimiraju odgovarajućim funkcijama. Što je manja granulacija diskretizacije, rezultati će biti bliži stvarnim rješenjima.

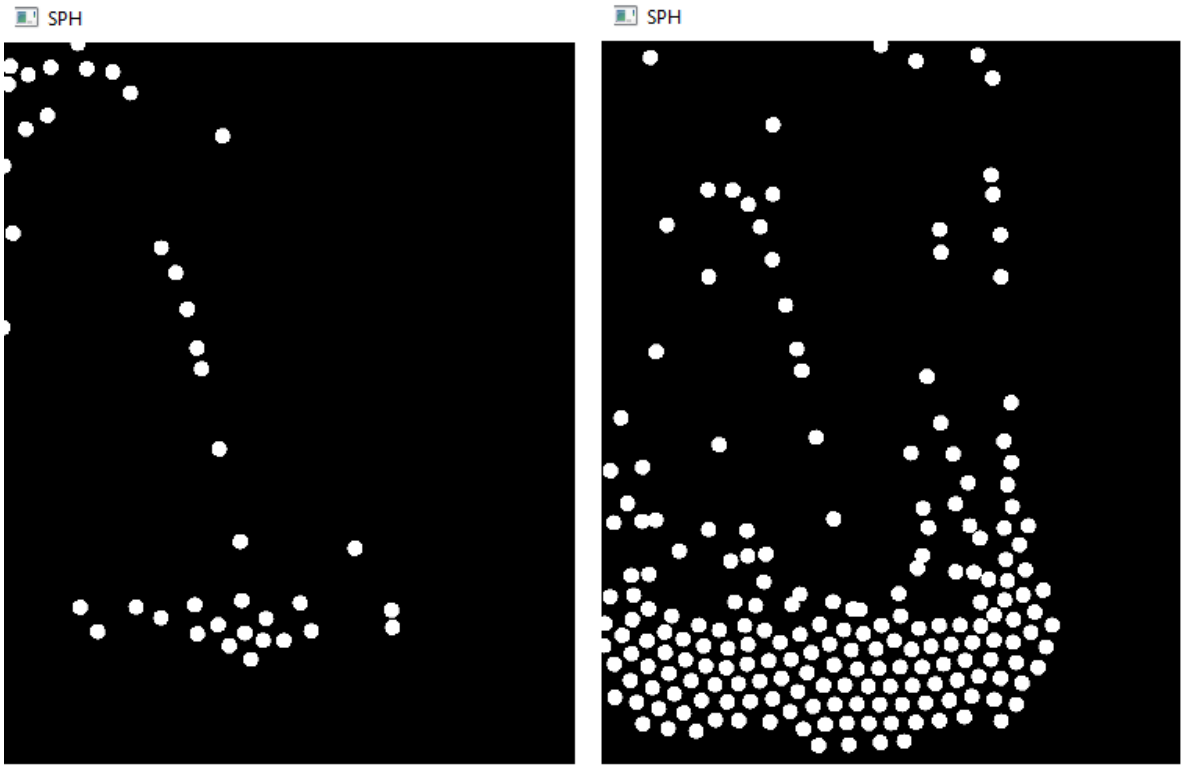
6.7. Simulacije

Napravljene su dvije simulacije. Prva prikazuje kvadrat fluid koji se počinje gibati pod utjecajem sile teže. Rezultat se može vidjeti na slici 6.3.



Slika 6.3 Prikaz prve simulacije prije i poslije

Druga simulacije počinje sa praznim ekranom. U svakom koraku glavnog algoritma na ekranu se generira jedna čestica fluida. Rezultat je na slici 6.4.



Slika 6.4 Prikaz druge simulacije prije i poslije

Zaključak

Kao znanost kompjutorska dinamika fluida još je uvijek u velikom razvoju. Teško je ne primijetiti da je kroz povijest njezin razvoj bio ograničen poznavanjem diferencijalnih jednadžbi. Najbolje rješenje i najbolji opis uvijek je bio jednak ili lošiji od najboljeg postupka baratanja i rješavanja diferencijalnih jednadžbi.

Simulacija fluida nije nimalo naivan i jednostavan postupak. Njegova realnost ide nauštrb vremena i računalnih resursa. Čak je i za jako snažna računala problematična. Potrebno je razumjeti da danas još uvijek ne možemo imati sve. Sa snagom današnjih računala i tehnikama koje se koriste nije moguće postići realističnu, fizikalno točnu, vizualno privlačno animaciju fluida koja se izvodi i renderira u realnom vremenu. Ključno je, stoga, na početku identificirati razloge zašto uopće simuliramo fluid. Ako je riječ o simulaciji za igru, sigurno će naglasak biti na izvođenju u realnom vremenu. Simulacija filmskih efekata će, s druge strane, zahtijevati što ljepši izgled bez obzira na vrijeme koje je potrebno utrošiti da bi se dobilo. Ako se pak želi simulirati rad motora i protok fluida u njemu, imperativ će imati fizikalna točnost. Svaki put kad pojačamo neku komponentu simulacije i damo joj veću važnost, to će biti nauštrb ostalih.

Literatura

[1] Andreić, Ž. (2014). *Temelji mehanike fluida.* Zagreb: Sveučilište u Zagrebu - Rudarsko- geološko- naftni fakultet.

[2] Ashford, O. (1985). Proheptor Profesor: The Life and Work of L. E. Richardson.

[3] Braley, C., i Tech. (n.d.). Fluid Simulation For Computer Graphics: A Tutorial in Grid Based and Particle.

[4] Bridson, R., & Muller-Fischer, M. (2007). Fluid Simulation SIGGRAPH 2007 Course Notes.

[5] Harada, T., Koshizuka, S., i Kawaguchi, Y. (n.d.).

[6] JiJi, V. (n.d.). CSCI 480 Computer Graphics.

[7] McDonough, J. M. (2009). Lectures in elementary fluid dynamics: Physics, Mathematics and Applications. Lexington.

[8] McKee, S., Tome, Tome, M., Ferreira, V., Cuminato, J., Castelo, A., Mangiavacchi, N. (2007). *The MAC method.*

[9] Nataesescu, V., i Gavrila, L. (2011). *About smoothing functions used in SPH methods.* Brasov.

[10] Šavar, M., Virag, Z., i Džijan, I. (2014). *Mehanika fluida (Skripta).* Zagreb.

[11] Shrainer, D., Sellers, G., Kessenich, J., i Licea-Kane, B. (n.d.). *OpenGL®Programming guide, Eight Edition.* Michigan.

[12] Ćupić, Marko i Mihajlović, Željka *Interaktivna računalna grafika kroz primjere u OpenGL-u*, 2016 , http://www.zemris.fer.hr/predmeti/irg/knjiga.pdf

[13] Bao, Y. B., Meskas J., *Lattice Boltzman Method for Fluid Simulation*, 2011.

Sažetak

Naslov: Simulacija fluida sustavom čestica

Najprije su proučena osnovna fizikalna svojstva fluida te jednadžbe po kojima se fluid giba. S obzirom da je za rad korišten i OpenGL proučene su njegove osnove. Rad se pozabavio trima osnovnim metodama kojima se postiže animacija fluida, a koje su izvodive u realnom vremenu. Prva je bila jednostavni sustav čestica bez interakcijskih sila među pojedinim česticama. Sljedeća je bila aproksimacijska visinska mapa koja je zatim integrirala i prvu metodu u svoj rad. Na kraju je implementiran jednostavni SPH simulator.

Ključne riječi: fluid, SPH, viskoznost, gustoća. tlak, čestica, visinska mapa, kernel funkcija, zaglađivanje

Summary

Title: Fluid simulation using particle system

Firstly basic physical attributes of fluids have been observed. Differential equations that describes fluid motion were also took in consideration. Considering the fact that OpenGL have been used for solution programming, the basic concepts of that programming language were described. There are there three methods that were implemented and described. Each one can run in real time. First method was simple particle system which doesn't calculate particle's interaction forces. The next one was heightmap. In the end simple SPH simulator was implemented.

Keywords: fluid, SPH, viscosity, density, pressure, particle, heightmap, kernel function, smoothing

Privitak

Programska potpora napravljena u sklopu teme diplomskog rada priložena je na CD-u.