SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD BR. 3200

ANIMACIJA TOKA FLUIDA

Tomislav Ljubej

Zagreb, lipanj 2013.

Sadržaj

1. Uvod

Fluidi su naravno vrlo raširenu u prirodi i susrećemo se svaki dan s njima pa je tako logično nastala želja za njihovim realističnim prikazom i simulacijom na računalima.

Pod pojmom fluidi ne govorimo samo o vodi i sličnim tekućinama koje su nestlačivi fluidi već i o plinovima koji mogu biti stlačivi. Stlačivi fluidi imaju sposobnost promjene volumena (odnosno gustoće), dok nestlačivi nemaju.

Iako je gibanje i interakcija fluida vrlo dobro opisana matematičkim jednadžbama, te jednadžbe su za bilo koji realni slučaj nesvedive na linearne već se mora uvesti određeni stupanj diskretizacije (a time i pogreške), tako se gotovo sve metode simulacije fluida danas prvenstveno bave raznima načinima diskretizacije tih jednadžbi kako bi s čim bržom simulacijom dobila čim veća uvjerljivost i preciznost rezultata. To nije nimalo jednostavno baš zbog složenog načina gibanja i svojstava fluida poput viskoznosti.

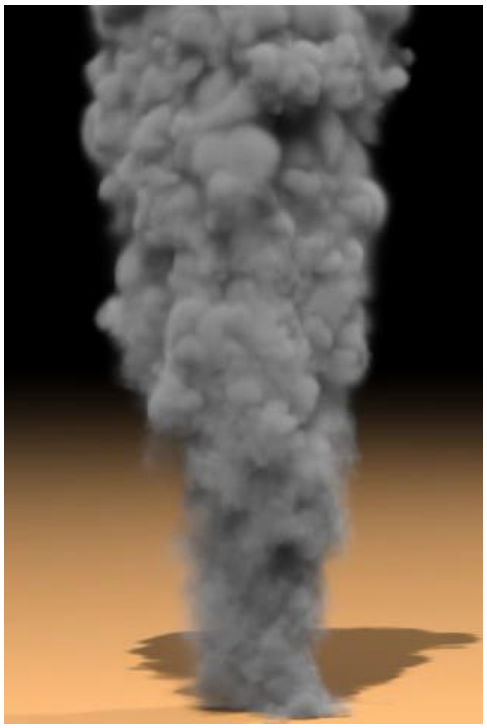
- *Slika 1. Simulacija vode*

Realistična i uvjerljiva simulacija fluida je već dugo predmet istraživanja u području računalne grafike ali je tek od sredine devedesetih godina prošlog stoljeća počela njena primjena u praksi, prvenstveno zbog toga što su tek tada računala postala dovoljno snažna za simuliranje u nekom razumnom vremenu. Simulacija fluida može se odvijati u realnom vremenu što naravno omogućuje i razne oblike

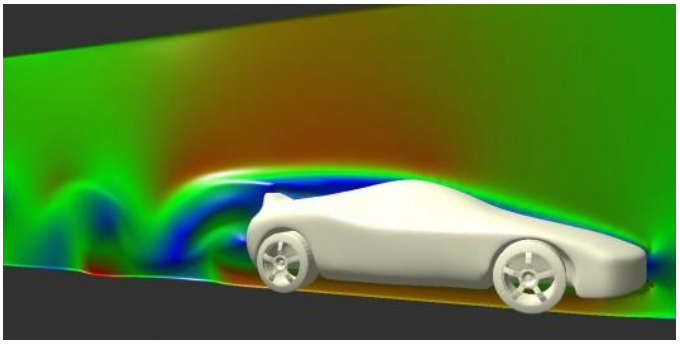
interakcije te može biti unaprijed izračunata pa onda prikazana u obliku videa, čime se naravno gubi mogućnost interakcije sa simulacijom.

Konkretne primjene su mnogobrojne, neke od njih su:

- Specijalni efekti u filmovima
- Dinamičniji dojam u video igrama (naravno uz uvjet da je simulacija moguća u stvarnom vremenu)
- Vizualizacija kretanja zraka oko krila aviona ili za analizu aerodinamičnosti automobila i drugih vozila



- *Slika 2. Simulacija dima.*



- Slika 3. Simulacija aerodinamičnosti automobila

2. Fizikalne osnove

2.1 Navier-Stokesove jednadžbe

Navier-Stokesove jednadžbe, nazvane prema Claude-Louise Navieru i George Gabriel Stokesu opisuju gibanje fluida kroz vrijeme te proizlaze iz primjene drugog Newtonovog zakona (F=ma) na gibanje fluida. One, za razliku od klasične mehanike gdje se opisuje položaj čestica u prostoru, ne opisuju direktno položaj fluida u prostoru već promjenu njegove brzine kroz vrijeme i prostor.

Zanimljivo je da Navier-Stokesove jednadžbe zbog svojih svojstava nemaju primjenu samo u područjima koja zahtijevaju simulacije fluida poput računalne grafike, meteorologije ili aerodinamike već i u područjima čiste matematike te ekonomije.

Razmatramo Navier-Stokesove jednadžbe za nestlačivi fluid. Fluid je u svakoj točki definiran vektorom brzine te tlakom u toj točki. Prva jednadžba za nestlačive fluide glasi:

$$\nabla \cdot \vec{u} = 0 \qquad (1)$$

Gdje vektor $\vec{u}$ predstavlja vektor brzine fluida. Ova jednadžba govori da u bilo kojem trenutku količina fluida koja utječe u neki volumen mora biti jednaka količini fluida koja istječe iz tog volumena, drugim riječima ova jednadžba osigurava nestlačivost našeg simuliranog fluida.

Druga jednadžba glasi:

$$\frac{\partial \vec{u}}{\partial t} = -\vec{u} \cdot \nabla \vec{u} - \frac{1}{\rho}\nabla p + \vartheta \nabla^2 \vec{u} + \vec{F} \qquad (2)$$

Gdje je $\frac{\partial \vec{u}}{\partial t}$ derivacija vektora brzine po vremenu, član $-\vec{u} \cdot \nabla \vec{u}$ proizlazi iz zakona o očuvanju količine gibanja te opisuje akceleraciju fluida u prostoru. Član $-\frac{1}{\rho}\nabla p$ opisuje sile kojima se fluid „opire“ pokušajima stlačivanja, ρ je gustoća fluida,

koja je u našem slučaju konstantna jer promatramo samo nestlačive fluide, a p je trenutni tlak fluida u nekoj točki. $\vartheta\nabla^2\vec{u}$ modelira viskoznost fluida, odnosno tendenciju čestica fluida da se kreću prosječnom brzinom susjednih čestica. Faktor ϑ je faktor viskoznosti, ako ga povećamo fluid će se doimati više „želatinastim“. Član $\vec{F}$ je zbroj bilo kakvih vanjska sila koja utječe na brzinu fluida u nekoj točki, obično je to gravitacija iako naravno mogu se uvesti i druge proizvoljne sile.

3. Modeli fluida

3.1 Metode rješavanja Navier-Stokesovih jednadžbi

Kao što je već prije spomenuto, Navier-Stokesove jednadžbe su nelinearne za veliku većinu slučajeva pa njihovom rješavanju je potrebno pristupiti na taj način. Nelinearne diferencijalne jednadžbe se u jako puno slučajeva ne mogu riješiti egzaktno osim ako nemaju određena povoljna svojstva, a pronalazak takvih rješenja se smatra značajnim otkrićem u području matematike. Iz tog razloga koristite se aproksimacije rješenja koje se obično dobiju nekakvim iterativnim postupkom.

Najčešće korištene metode su: Metoda konačnih razlika, metoda konačnih volumena te metoda konačnih elemenata.

3.2 Modeli fluida

Fluide možemo modelirati na dva glavna načina: eulerovski i lagrangianski.

3.2.1 Eulerovski model

Eulerovski model diskretizira fluid kao rešetku konačnog broja ćelija. U svakoj ćeliji je definirana brzina i tlak čestice. Pomoću rešetke procjenjuje se derivacija Navier-Stokesovih jednadžbi metodom konačnih razlika.

Tako onda divergenciju brzine fluida u proizvoljnoj ćeliji dobijemo na sljedeći način:

$$\nabla \cdot \vec{u} = u_x(x+1,y,z) - u_x(x,y,z) + u_y(x,y+1,z) - u_y(x,y,z) + u_z(x,y,z+1) - u_z(x,y,z) \quad (3)$$

Gdje su u_x, u_y, u_z x, y i z komponente brzine u tim točkama.

Divergenciju tlaka dobijemo na vrlo sličan način:

$$\nabla \cdot p = p(x+1,y,z) - p(x,y,z) + p(x,y+1,z) - p(x,y,z) + p(x,y,z+1) - p(x,y,z) \quad (4)$$

Laplasijan vektora brzine je sljedeći:

$$\nabla^2 \vec{u} = (\nabla^2 u_x(x,y,z), \nabla^2 u_y(x,y,z), \nabla^2 u_z(x,y,z)) \quad (5)$$

Gdje se na primjer $\nabla^2 u_x(x,y,z)$ računa kao:

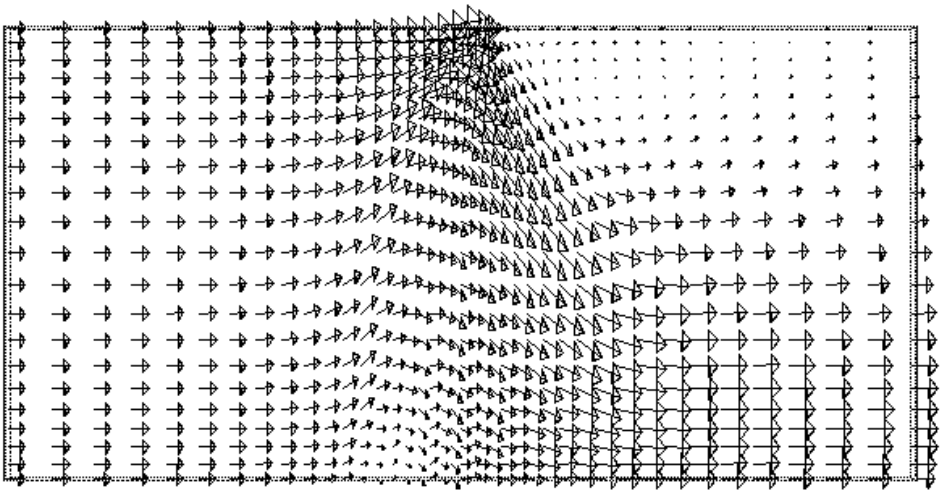
$$\nabla^2 u_x(x,y,z) = u_x(x+1,y,z) + u_x(x+1,y,z) + u_x(x,y+1,z) + u_x(x,y-1,z) + u_x(x,y,z+1) + u_x(x,y,z-1) - 6u_x(x,y,z) \quad (6)$$

Navedene formule vrijede za fluid u tri dimenzije međutim pojednostaviti ih na dvije dimenzije je trivijalno.

Prednost eulerovskog modela jest sposobnost detaljnog prikazivanja procesa koji se zbivaju unutar fluida pomoću raznih interpolacija, jednostavnije je izvesti očuvanje nestlačivosti fluida te jednostavno ga je paralelizirati.

Nedostaci su samo postojanje rešetke, ne možemo odrediti stanje fluida izvan rešetke no taj problem se može riješiti upotrebom na primjer *hash* tablice za spremanje ćelija umjesto fiksnog polja ćelija pa je onda moguće dinamički stvarati i uništavati ćelije po potrebi. Velik nedostatak je i prilično velika računska zahtjevnost ove metode.

Popularna varijanta za simulaciju fluida eulerovskog modela je MAC (Marker And Cell) metoda prvi puta prezentiran u radu Harlowa i Wencha *[1]* o kojoj će biti više rečeno kasnije te koju ćemo koristiti u ovome radu za implementaciju simulacije.



Slika 4. - Primjer prikaza eulerovskog modela fluida, svaka strelica predstavlja brzinu fluida u jednoj ćeliji

3.2.2 Lagrangianski model

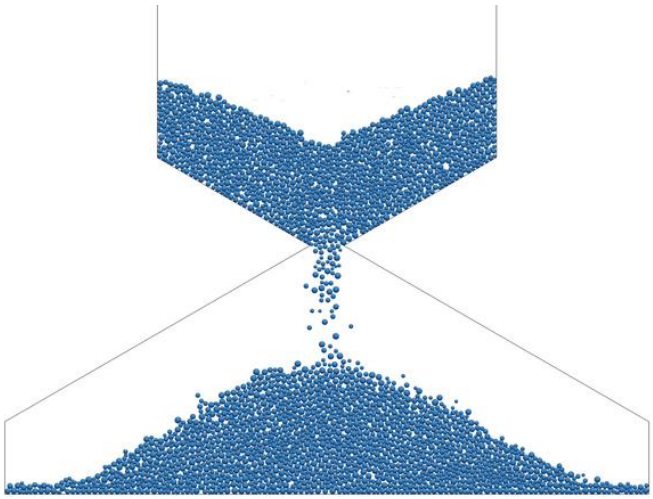
Lagrangianski model prikazuje fluid kao skup konačnog broja čestica koje se slobodno kreću po prostoru, dakle nisu ograničene rešetkom. Svaka čestica je opisana svojim položajem, brzinom te vektorom dobivenim sumiranjem svih sila koje djeluju na nju (na primjer gravitacija).

Modeliranje međudjelovanja tih čestica sa svrhom da se ponašaju poput fluida može se izvesti na više načina. Najčešće korištena metoda jest smoothed-particle hydrodynamics (SPH). Ona se temelji na skupu čestica čija svojstva (brzina, sile, tlak, itd.) se „zaglađuju“ određenim funkcijama sa svojstvima susjednih čestica ako su one unutar tzv. „radijusa zaglađivanja“. U simulaciji fluida po uzoru na Navier-Stokesove jednadžbe izračunavaju se utjecaj viskoznosti, tlaka, te naravno drugih sila poput gravitacije ili čvrstih objekata.

Prednosti ovog modela jest znatno manja računska zahtjevnost stoga se ovakva metoda više koristi u interaktivnim simulacijama i video igrama.

Glavni nedostatak je veća „grubost“ simulacije što onda ima za posljedicu manju uvjerljivost. Rješenje je naravno povećanje količine čestica međutim onda

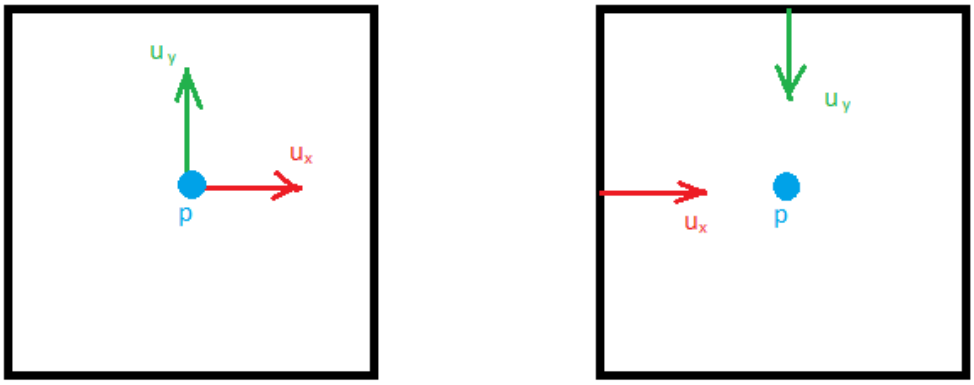
naravno znatno raste i računska kompleksnost. Još jedan nedostatak je što je očuvanje nestlačivosti fluida teže izvedivo jer se jednadžba za tlak ne razrješava na cijelom fluidu kao u eulerovskom modelu već samo unutar prije spomenutog „radijusa zaglađivanja“.



Slika 5. – Primjer lagrangianskog sustava, čestice se slobodno kreću prostorom, nisu ograničene rešetkom

3.3 Metoda MAC

Metoda MAC se, kao što je već prije spomenuto, temelji na eulerovskom modelu. Svaka ćelija je definirana s vektorom brzine u sredini ćelije te tlaku također u sredini ćelije. Komponente brzine, umjesto u sredini ćelije, mogu biti definirane i na stranicama kocke (ili kvadrata u 2D slučaju) koji predstavljaju ćeliju, što daje precizniju i stabilniju simulaciju iako otežava interpolaciju brzine. Globalno se definira i viskoznost fluida te vanjske sile u pojedinim ili svim ćelijama, obično je to samo gravitacija.



Slika 6. – Lijevo je primjer 2D ćelije s brzinama definiranim u sredini a na desno je brzina definirana na stranicama ćelije, tlak je uvijek definiran u centru ćelije

Sama rešetka može biti implementirana kao fiksno dvodimenzionalno polje ćelija što naravno jako ograničava mogući prostor simulacije, međutim možemo koristiti i *hash* mapu čime onda vrlo lako možemo po potrebi dodavati i uništavati ćelije, te simulacija više nije toliko prostorno ograničena.

Svaka ćelija isto tako ima i definirano stanje, moguća stanja su „fluid“, „prepreka“ i „zrak“.

Uz samu rešetku definiran je i skup čestica koje su zapravo točke u kontinuiranom prostoru, dakle nisu diskretizirane kao rešetka. One služe kako bi se kroz simulaciju pratio položaj fluida, dakle svaka ćelija koja sadrži barem jednu česticu će biti označena sa stanjem „fluid“. Iako se može nekome činiti da se ovdje radi o lagrangianskom modelu pošto imamo čestice koje se slobodno gibaju po prostoru to nije tako jer ove čestice služe isključivo kao pomoć pri određivanju gdje se fluid nalazi u prostoru, one ne sudjeluju u nikakvim proračunima simulacije. Valja napomenuti da broj čestica nije jednak broju ćelija, obično se na početku simulacije u svaku ćeliju fluida postavi 4 čestice.

Simulacija se odvija u diskretnim pomacima od Δt vremena, s tim da valja napomenuti da Δt nemora nužno biti konstantan kroz cijelu simulaciju.

Sama simulacija se odvija u više koraka. Na početku se određuje koje će se ćelije označiti kao „fluidi“, a brzina svih ćelija koje nisu označene kao fluid se postavlja na 0.

Nakon toga se odvija advekcija brzina, što ukratko znači da trenutna brzina fluida u nekoj točci ovisi o brzini i položaju u kojoj se taj fluid nalazio u prošlom koraku simulacije. Ovaj postupak daje simuliranom fluida onu „vrtložnost“ koju vidimo i u pravim fluidima.

Nakon toga se odvija izračun efekata viskoznosti na način da se svakoj ćeliji njena trenutna brzina „uskladi“ s brzinama susjednih ćelija s nekim faktorom ϑ. Što je faktor ϑ veći to će efekt viskoznosti biti veći, tj. susjedne brzine će se manje razlikovati a fluid će se doimati „želatinastijim“.

Nakon ovih postupaka fluid koji smo dobili nije nužno nestlačiv, pa tako moramo izvršiti postupak očuvanja nestlačivosti. To radimo tako da izračunamo tlak u svakoj ćeliji zasebno te modificiramo brzinu svake ćelije temeljem razlika tlakova između susjednih ćelija.

Sada ekstrapoliramo brzine na granicama fluida u proizvoljno široko područje oko fluida.

Naposljetku pomičemo čestice temeljem novo izračunatih brzina te ponavljamo cijeli postupak u sljedećoj iteraciji.

Svaki od ovih koraka se može izvršiti na znatno različite načine te je puno novih dostignuća na području simulacije fluida su upravo pronalazak bržeg ili stabilnijeg načina za izvršavanje nekog od ovih koraka.

4. Implementacija Metode MAC

U ovom radu ćemo implementirati jednostavniji oblik Metode MAC. Simulacije će se odvijati u dvije dimenzije zbog manje računske zahtjevnosti Implementacija je izvedena u programskom jeziku C++. Vizualizacija rezultata je izvedena pomoću programskog sučelja OpenGL s bibliotekom SFML kao potporom za lakše kreiranje prozora i procesiranje interakcije s korisnikom. Vremenski pomak Δt će biti konstantan kroz cijelu simulaciju te neće ovisiti o stvarnom vremenu potrebnom za računanje i prikaz jednog koraka simulacije, dakle prividna brzina toka simulacije će ovisiti o brzini računala koje izvodi simulaciju.

Animacija je napravljena tako da svaki korak simulacije ponovo iscrtamo sve čestice metodom OpenGL-a GL_POINTS, a prepreke iscrtavamo metodom GL_QUADS.

Za ostvarenje interaktivnosti korištena je biblioteka SFML, svaki korak simulacije gleda se da li je pritisnuta određena tipka miša ili tipkovnice te se obavljaju određene akcije ako jest. Korisnik tako može na primjer lijevim klikom miša dodati novi komadić fluida u prostor simulacije, desnim klikom miša dodati novu prepreku, a tipkom 'R' izbrisati cijeli fluid s ekrana.

Kod je većim dijelom baziran na radu Jos Stama *[2]*.

4.2 Strukture podataka

Rešetka je definirana fiksnim dvodimenzionalnim poljem ćelija veličine *(N+2)*(N+2)*, gdje se *N* zadaje u glavnom programu. *N* određuje rezoluciju simulacije. Razlog zašto smo dodali još 2 sloja ćelija na strane rešetke je taj što će te dodatne rubne ćelije predstavljati „zidove“ naše simulacije te se kroz njih uglavnom neće iterirati nego će samo sudjelovati u nekim proračunima. Globalno je još definirana viskoznost fluida te vektor vanjske sile koja djeluje na sve ćelije fluida, u konkretnom slučaju ta sila je gravitacija.

Za svaku ćeliju je definirana brzina na x i na y osi te tlak u obliku broja s pomičnim zarezom dvostruke preciznosti. Također za svaku ćeliju je u obliku boolean vrijednosti definirano da li je ćelija tipa „fluid“ ili tipa „prepreka“, a ako nije ni jedno

uzima se da je tipa „zrak“. Još je definirana i kopija prijašnjih vrijednosti brzina jer za neki od postupaka algoritama ne možemo brzine promijeniti samo na osnovi trenutnih vrijednosti. Definirana je i cijelobrojna vrijednost broja iteracija u određenim koracima algoritma gdje je potrebno aproksimirati rješenje velike linearne jednadžbe. Naposlijetku je definirano polje čestica koje određuju gdje se nalazi fluid.

Brzine se uzimaju da su definirane u centru ćelije.

```
class FluidSimulator
{
      …
    double *u_vel, *v_vel; // u=x os, v=y os,
    double  *u_prev, *v_prev; // u=x os, v=y os, kopije starih
vrijednosti
    double visc // viskoznost
    double dt; // vrijeme izmedju koraka simulacije
    double gx, gy; // vanjska sila (gravitacija) na x i y osi
    int niter; // broj iteracija za neke korake algoritma
    int N; // rezolucija simulacije
    char *fluidcells; // polje gdje je zapisano da li neka celija
pripada fluidu
    char *solidcells; // isto ali za prepreke
    std::vector<Vec2> markers; // polje cestica koje odredjuju
fluid
      …
};
```

Slika 7. – dio koda koji definira potrebne vrijednosti za izvođenje simulacije

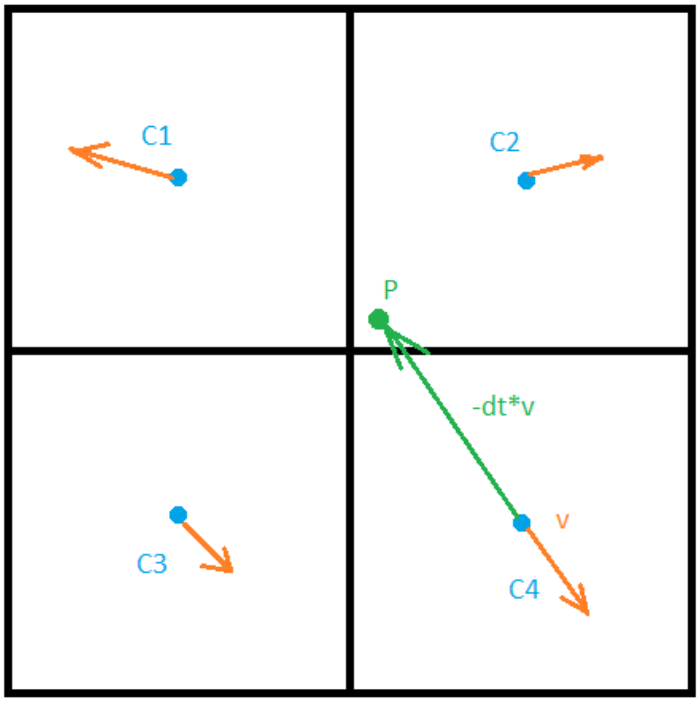
4.3 Algoritam

Na početku se vrši advekcija brzina svih ćelija. To radimo tako da za svaku ćeliju fluida se iz sredine ćelije pomaknemo za $-\Delta t\vec{v}$ gdje je $\vec{v}$ vektor brzine ćelije te očitamo vrijednost vektora brzine u novodobivanoj točki tako da napravimo bilinearnu interpolaciju između brzina četiri ćelije između kojih se nalazi dobivena točka. Za interpolaciju koristimo stare vrijednost brzina jer bi inače novoizračunate vrijednosti za neku ćeliju moguće utjecale na račun brzine neke druge ćelije. Naposlijetku zamijenimo pokazivače na stare vrijednosti brzina s „pravim“

vrijednostima kako bi idući korak algoritma mogao koristiti upravo izračunate vrijednosti. Funkcija je u kodu definirana na sljedeći način te je pozivamo dva puta, jednom za x-os a jednom za y-os os:

```
void advect(int b, double *d, double *d0, double *u, double *v);
```

Gdje je b parametar koji je 1 ako radimo advekciju s brzinama na x-osi a 2 ako radimo na y-osi. Parametri *d* i *d0* će biti *u_vel* i *u_prev* ili *v_vel* i *v_prev* ovisno za koju os računamo a parametri *u* i *v* će biti fiksno *u_prev* i *v_prev*.



Slika 8. – Određujemo novu brzinu za ćeliju C4, brzina u točki P se dobiva bilinearnom interpolacijom brzina između točaka C1, C2, C3 i C4

Nakon advekcije slijedi primjena efekta viskoznosti. To možemo raditi samo tako da od četiri susjedne ćelije oduzmemo gornju od donje brzine te lijevu od desne, pomnožimo s faktorom viskoznosti te dodamo ih na trenutno računanu ćeliju. No međutim to će simulaciju činiti prilično nestabilnom za bilo koji malo veći faktor viskoznosti zato što je cijeli postupak zapravo veliki linearni sustav gdje su nepoznanice nove brzine. Taj sustav se može riješiti egzaktno međutim to u našem slučaju nije nužno, u simulaciji je rješenje aproksimirano Gauss-Seidolovom metodom koja funkcionira tako da kroz određeni broj iteracija (obično oko 100 je

dovoljno) jednostavnim aritmetičkim operacijama aproksimira pravo rješenje svih varijabli. Postupak opet provodimo samo na ćelijama fluida.

Funkcija je u kodu definirana na sljedeći način te je opet pozivamo zasebno za x-os i y-os:

```
void apply_viscosity(int b, double *x, double *x0, double visc);
```

Gdje *b* ima istu svrhu kao i u funkciji „advect“, *x* i *x0* su *u_vel* i *u_prev* za slučaj kada računamo viskoznost za x-os te *v_vel* i *v_prev* za y-os. Parametar visc je faktor viskoznosti.

Sada radimo izračun tlaka u svakoj ćeliji fluida. To radimo na vrlo sličan način kao i u proračunu viskoznosti jer se radi o sličnom linearnom sustavu, dakle opet koristimo Gauss-Seidelovu metodu. Prvo u svim ćelijama vrijednost tlaka postavimo na nula. Onda u svakoj ćeliji fluida izračunam divergenciju brzine prema formuli (3) koju onda koristimo za izračun tlaka.

Nakon izračunatih tlakova treba primijeniti silu koji oni stvaraju na brzine ćelija. To radimo tako da za svaku ćeliju oduzmemo tlak lijevog i desnog te gornjeg i donjeg susjeda, pomnožimo s Δt te pridodamo negiranu vrijednost rezultata trenutnoj brzini u ćeliji, time „guramo“ brzinu od područja većeg tlaka. Postupak možemo prikazati i sljedećim formulama:

$$u'_x(x,y) = u_x(x,y) - \Delta t \cdot (p(x+1,y) - p(x-1,y)) \quad (6)$$

$$u'_y(x,y) = u_y(x,y) - \Delta t \cdot (p(x,y+1) - p(x,y-1)) \quad (7)$$

Nakon ovog postupka u fluidu je napokon osigurana nestlačivost. Definicija funkcije je sljedeća:

```
void project(double *u, double *v, double *p, double* div);
```

Gdje su *u* i *v* brzine ćelija, *p* tlakovi, a *div* bilo koje privremeno polje u koje spremamo divergencije brzina ćelija.

Sada vršimo ekstrapolaciju brzina fluida u okolne ćelije zraka. To radimo na vrlo jednostavan način, za svaku ćeliju zraka koja je susjedna ćeliji fluida brzinu postavimo na prosjek svih susjednih ćelija te tu ćeliju privremeno označimo kao

ćeliju fluida te cijeli postupak ponavljamo nekolicinu puta tako da imamo nekoliko ćelija debeli sloj ćelija oko fluida s ekstrapoliranim brzinama. Ovo omogućava česticama da se nešto realističnije kreću.

Još valja napomenuti da nakon svakog postupka u kojem mijenjamo brzine ćelija trebamo paziti na brzine u ćelijama prepreka i zidova simulacije. To rješavamo vrlo jednostavno, ideja je da horizontalna brzina na rubovima horizontalnih zidova bude jednaka nuli. To postižemo jednostavno tako da horizontalnu komponentu brzine (*u_vel*) postavimo da bude jednaka negativnoj vrijednosti brzini susjedne ćelije koja nije zid ili prepreka. Isto naravno vrijedi i za vertikalne komponente brzine. Funkcija je definirana na sljedeći način:

```
void set_bnd(int b,double* x);
```

Gdje je *b* jednak 1 ako se radi o x-osi te 2 ako se radi o y-osi. Ovu funkciju pozivamo između ostalog i u funkcijama *advect* i *apply_viscosity*, zato nam je u njima trebao parametar *b*.

Naposlijetku pomičemo sve čestice tako da za svaku prvo nađemo brzinu u točki u kojoj se ona nalazi pomoću bilinearne interpolacije isto kao i u postupku advekcije te onda jednostavno eulerovskom integracijom pomaknemo česticu za $\Delta t\vec{v}$.

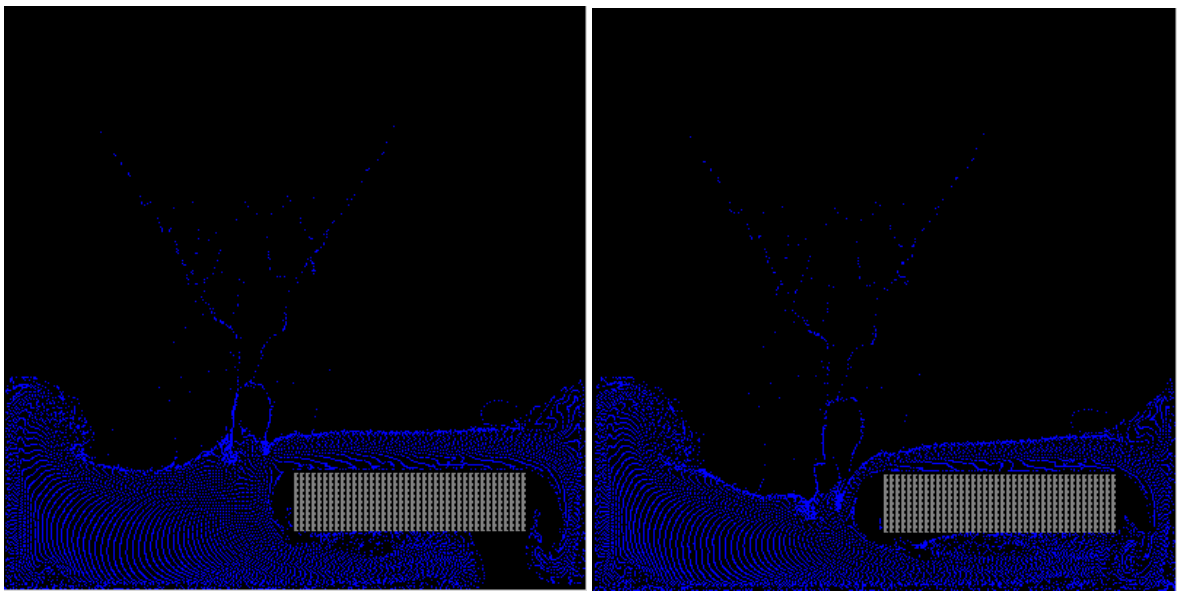
4.4 Testiranje simulacije

Implementacija simulacije je napravljena da ima određene mogućnosti interakcije u stvarnom vremenu pa tako je potrebno odabrati parametre simulacije koji će učiniti simulaciju manje računski zahtjevnom kako bi se mogla odvijati u stvarnom vremenu. Tako za vrijednost rezolucije rešetke *N* odabiremo vrijednost 100, za broj iteracija za Gauss-Seidelovu metodu uzimamo također 100, a za vrijednost vremenskog pomaka *dt* uzimamo 0.03. Ove vrijednosti daju razumno precizno simulaciju ali svejedno omogućuju izvođenje u stvarnom vremenu.

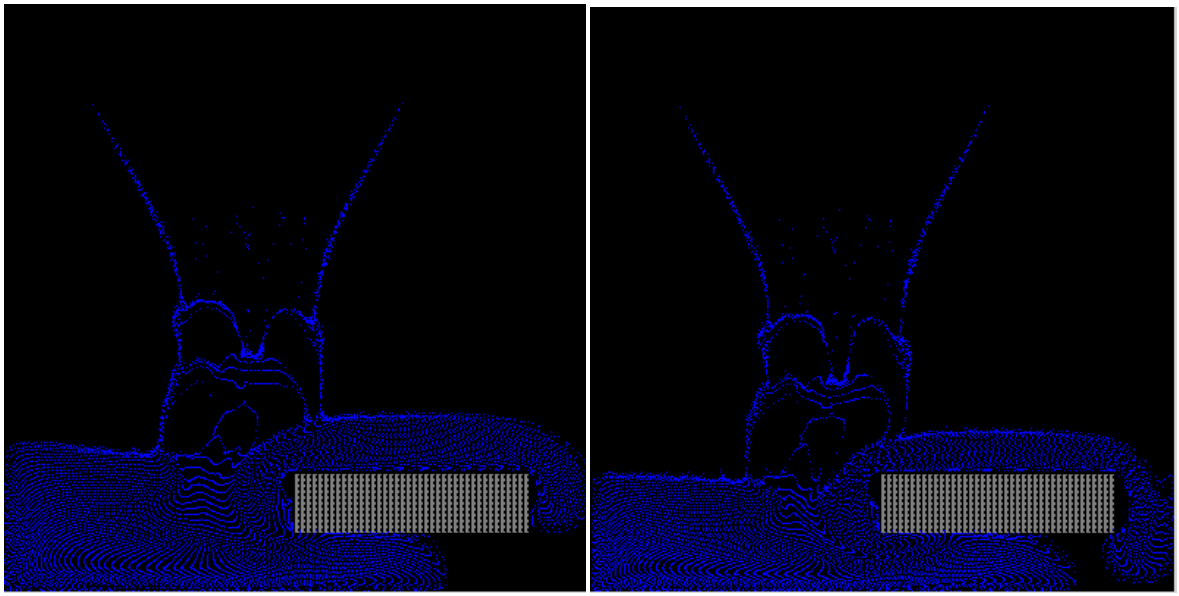
Ako želimo da se fluid ponaša kao voda za vrijednost faktora viskoznosti uzmemo nula. Za vrijednosti jednake i veće od 0.001 fluid će se ponašati poput npr. sredstva za pranje posuđa, nećemo više primjećivati valove na površini fluida. Vrijednosti

veće od 0.2 uzrokuju vrlo nerealnu simulaciju te postoje posebne tehnike za simuliranje fluida vrlo visoke viskoznosti.

Povećanjem gravitacije ne dobijemo znatno drugačije ponašanje, fluid jednostavno brže padne na dno simulacije i brže se prestane gibati.



Slika 9. – Primjer izvođenja simulacije s faktorom viskoznosti 0, fluid se ponaša slično kao voda



Slika 10. – Primjer izvođenja simulacije s faktorom viskoznosti 0.003, u fluidu više ne vidimo valove te se fluid sporije širi po dnu prostora

4.5 Moguća poboljšanja i nadogradnje simulacije

Preciznost simulacije možemo poboljšati na više načina čak i da ne izgubimo na brzini izvođenja.

Kao što je već rečeno preciznost simulacije se može poboljšati tako da se komponente brzine ćelija definiraju na stranicama ćelija umjesto u sredini ćelije, međutim to prilično zakomplicira interpoliranje brzine u proizvoljnoj točki te moramo dodatno paziti na to da li neka komponenta brzine graniči s određenim tipom ćelije.

Isto tako preciznost simulacije možemo poboljšati tako da advekciju te pomicanje čestica ne radimo s eulorovskom integracijom nego integracijom Runge-Kutta reda dva (RK2), ona u pomicanje čestice uzima u obzir i brzinu koja se nalazi na pola puta od promatrane točke pa do punog eulerovskog pomaka za $\Delta t\vec{v}$ koju dobijemo tako da se od čestice pomaknemo za $\Delta\frac{t}{2}\vec{v}$. Formula je sljedeća:

$$\vec{x}' = \vec{x} + \Delta t\vec{v}(\vec{x} + \Delta\frac{t}{2}\vec{v}(x)) \tag{8}$$

Gdje je $\vec{x}'$ nova pozicija, $\vec{v}(x)$ brzina u trenutnoj poziciji, a $\vec{v}(\vec{x} + \Delta\frac{t}{2}\vec{v}(x))$ brzina na pola puta punog eulerovskog pomaka.

Simulaciju možemo potencijalno ubrzati ako koristimo dinamički način određivanja vremenskog pomaka Δt. To radimo tako da prvo pronađemo najveću apsolutnu vrijednost brzine u sustavu te izračunamo Δt prema formuli:

$$\Delta t = \frac{|\vec{v}_{max}|}{\frac{1}{N}} \tag{9}$$

Gdje je $|\vec{v}_{max}|$ najveća apsolutna vrijednost brzine u sustavu, a $\frac{1}{N}$ je širina jedne ćelije. Drugim riječima ovime osiguravamo da najbrža čestica nikada neće „preskočiti“ jednu cijelu ćeliju u jednom koraku simulacije. Možemo uzeti i nešto veću vrijednost od ove međutim onda trebamo raditi neke druge dijelove simulacije poput advekcije znatno drugačije.

Simulacija se dodatno može nadograditi s na primjer mogućnošću simulacije interakcije s pokretnim krutim objektima međutim to zahtjeva znatno kompleksne postupke te vrlo dobro poznavanje fizike i matematike.

Navedena poboljšanja opisana su u radu Clinea, Cardona i Egberta *[3]*.

Može se dodati i podrška za više tipova fluida koji se razlikuju po viskoznosti, to bi se moglo napraviti tako da dodamo više tipova ćelija, na taj način bi onda imali tipove ćelija „fluid 1“, „fluid 2“ itd. ovisno koliko tipova fluida želimo u simulaciji. No međutim onda se javlja problem kako na uvjerljiv način implementirati miješanje različitih tipova fluida.

Određeni koraci simulacije poput advekcije i pomicanje čestica se mogu prilično jednostavno paralelizirati što bi dodatno ubrzalo simulaciju.

5. Zaključak

Simulacija fluida u računalnoj grafici je vrlo veliko i kompleksno područje. Ovaj rad je bio samo kratak uvod u moguće metoda simulacije, kao što je već rečeno moguća poboljšanja i nadogradnje simulacije su mnogobrojne i još uvijek se pronalaze nove i bolje metode.

Metoda simulacije koja je korištena u ovom radu (metoda MAC) je dosta precizna ali zbog svoje računske zahtjevnosti neprimjerena za simulacije u stvarnom vremenu. U ovom radu je to bilo moguće zbog relativno malo prostora simulacije i zbog toga što se radi o 2D fluidu, a čak i tada je cijela simulacija djelovala pomalo usporeno iako su naravno moguće daljnje optimizacije implementacije u ovom radu. Za simulaciju u stvarnom vremenu je primjerenija metoda SPH.

Sam postupak implementacije nije jednostavan jer je metoda simuliranja po svojoj prirodi vrlo osjetljiva, mora se paziti na mnogo implementacijskih detalja i rubnih slučajeva te čak i najmanja pogreška u implementaciji može dovesti do vrlo nerealne ili nestabilne simulacije. Naravno potrebno je paziti da se pravilno odaberu parametri simulacije poput rezolucije simulacije, viskoznosti, sile gravitacije itd. jer i oni puno utječu na uvjerljivost i stabilnost simulacije.

6. Sažetak

U ovom radu napravljen je kratak uvod u razne metode simulacija fluida. Opisane su Navier-Stokesove jednadžbe za nestlačive fluide te osnovni principi rada eulerovske i lagrangianske metode. Detaljnije je prezentirana eulerovska metoda MAC te je opisana jednostavna implementacija te metode u programskom jeziku C++ i sučelju OpenGL. Opisani su rezultati izvođenja simulacije s različitim vrijednostima parametera te su na kraju opisana neka moguća poboljšanja simulacije u svrhu bržeg ili preciznijeg izvođenja.

Ključne riječi: fluidi, simulacija fluida, metoda MAC, jednadžbe Navier-Stokes, animacija fluida

7. Abstract

This paper describes various methods for fluid simulation. Basic principles of the Navier-Stokes equations for incompressible fluids and both Eulerian and Lagrangian simulation methods are described. The MAC method and a simple implementation of the method in C++ and the OpenGL API are further described in greater detail. The simulation is then tested with varying parameters and some suggestions for improvement of simulation stability and speed are presented.

Keywords: fluids, fluid simulation, MAC method, Navier-Stokes equations, fluid animation

8. Literatura

1. *Harlow F. H. i Welch J. E. Numerical calculation of time-dependant viscous incompressible flow of fluid with a free surface. The Physics of Fluids* 8. 1965. 2182-2188
2. Jos Stam, GDC03.pdf, *Real Time Fluid Dynamics for Games*, http://www.dgp.toronto.edu/people/stam/reality/Research/pdf/GDC03.pdf, 15.05.2013
3. David Cline et. al., fluidFlowForTheRestOfUs.pdf, *Fluid Flow for the Rest Of Us: Tutorial of the Marker and Cell Method in Computer Graphics, http://people.sc.fsu.edu/~jburkardt/pdf/fluid_flow_for_the_rest_of_us.pdf, 25.05.2013*