SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 2988

ANIMACIJA FIZIKALNO TEMELJENOG MODELA VATRE

Teon Banek

Zagreb, svibanj 2013.

Hvala roditeljima na podršci i mentoru prof. dr. sc. Željki Mihajlović na pomoći u izradi ovog rada.

Sadržaj

1. Uvod

Vatra je oduvijek fascinirala ljude. Njeni pozitivni efekti poput grijanja i osvjetljenja nam omogućuju lakši i bolji život ali s druge strane nalaze se i opasnosti vatre o kojima nažalost čujemo na vijestima ili čitamo u novinama.

Prekretnica u životu ljudi je bilo ostvarenje mogućnosti kontrole vatre te njeno kontroliranje u svijetu računarstva je postalo od izrazite važnosti. Računalno simulirana vatra se koristi u različite svrhe a neke od njih su:

- vizualni efekti gorenja i eksplozija u filmovima,
- grafička reprezentacija u video igrama,
- proučavanje ponašanja vatre u nekom stvarnom prostoru kako bi se bolje odredio protokol evakuacije.

Za različite svrhe koriste se različiti načini računalne simulacije, koje nude različite stupnjeve detaljnosti simulacije i korištenja računalnih resursa. Kod vizualnih efekata u igrama želimo brzo izvođenje simulacije pa njeno pravilno ponašanje je od manje važnosti nego brzina algoritma. Za filmove i znanstvene simulacije možemo dopustiti trajanje algoritma u danima kako bi dobili što precizniji i stvarniji model vatre.

Ovaj završni rad će dati kratki pregled nekoliko metoda simulacije vatre te pokazati implementaciju jedne takve metode. Temelj fizikalnog opisa vatre počiva na tome što vatru možemo opisati kao poseban slučaj fluida, pa sve metode polaze od računalne simulacije fluida.

2. Računalna dinamika fluida

Računalna dinamika fluida (eng. *Computational Fluid Dynamics*) je grana znanosti koja se bavi numeričkim metodama i algoritmima u rješavanju mehanike fluida. Termin fluid se često koristi kao sinonim za tekućine na kojima najlakše možemo vidjeti svojstva fluida: protjecanje odnosno deformiranje uslijed sila trenja. Pod pojam fluida ulaze još plinovi i plazme. Ponašanje fluida je opisano skupom jednadžbi parcijalnih derivacija koje zajednički zovemo Navier-Stokesove jednadžbe. Njihovi izračuni su još uvijek zahtjevni za današnja računala pa se koriste varijante tih jednadžbi za nestlačive fluide čak i kada se modelira stlačivi fluid poput plina. Popularne tehnike u simulaciji fluida se mogu podijeliti u dvije grupe. Jedna grupa su tehnike bazirane na česticama a druga grupa je bazirana na uzorkovanju fluida unutar rešetke odnosno polja. Predstavnici, opisani u nastavku, za svaku grupu, respektivno su: metoda hidrodinamike zaglađenih čestica (eng. *Smoothed Particle Hydrodynamics*, *SPH*) i metoda mreže markera i ćelija (eng. *Marker And Cell grid*, *MAC grid*).

2.1 Metoda Smoothed Particle Hydrodynamics (SPH)

SPH metoda pripada u tehniku simulacije fluida baziranoj na simulaciji čestica (eng. *particle*) te je nadogradnja na jednostavan sustav čestica (eng. *Simple Particle System*). Takav sustav se zove jednostavnim jer ne modelira interakcije među česticama. Svaka čestica sadrži osnovne podatke o sebi:

- masu,
- poziciju u prostoru
- vektor brzine,
- vektor akumuliranih sila koje djeluju na česticu te
- životni vijek (ako je potrebno nestajanje nakon nekog vremena).

Čestice kreira izvor (eng. *emitter*) u određenim intervalima te im daje početnu brzinu i poziciju. Formule koje određuju gibanje čestica su jednostavne diferencijalne jednadžbe ovisnosti puta o brzini te akceleraciji o sili i masi.

Takav način omogućuje jednostavan i brz račun dinamike velikog broja čestica.

SPH nadograđuje takav sustav dodavanjem međudjelovanja čestica te zaglađivanjem uzorkovanja čestica u prostoru kako bi se uklonio grub prikaz fluida. Zaglađivanje je ostvareno težinskom funkcijom koja ovisi o udaljenosti čestica kako bi se odredila gustoća fluida u prostoru. Međudjelovanje čestica kreće iz Navier-Stokesovih jednadžbi, prema kojima se u ukupan izračun sile uključuje računanje viskoznosti fluida, tlak oko čestica te dodatne sile poput gravitacije. Brzina izračuna je i dalje na zadovoljavajućoj razini te je to i jedna od glavnih prednosti ove metode.

Prikazivanje takvog fluida se izvodi interpolacijom različitih gustoća što daje ljepši ali računalno zahtjevniji izgled, a najčešća tehnika je prikazivanje malih slikovnih elemenata (eng. *sprite*) na pozicijama čestica što se izvodi jako brzo i efikasno ali na štetu izgleda. Primjer ove metode koja koristi slikovne elemente za prikaz nalazi se na slici 1.

Slika 1. Vatra ostvarena sustavom čestica u igri S.T.A.L.K.E.R. - Clear Sky

Glave prednosti metode *SPH* su:

- garancija očuvanja mase jer svaka čestica pamti svoju masu,
- izračuni jednadžbi su brzi i efikasni te
- lako dodavanje dodatnih sila koje utječu na čestice.

Glavni nedostaci su:

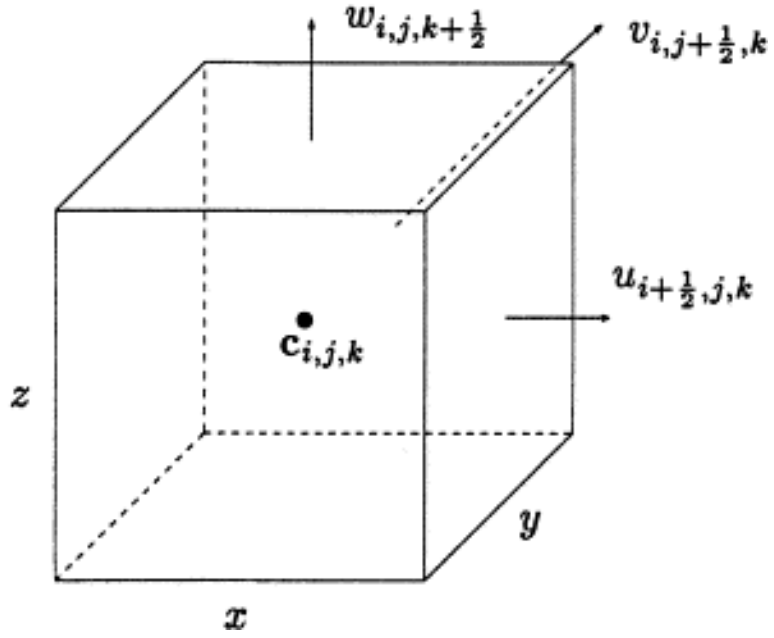
- potreban broj čestica naglo raste ako je potrebna veća razlučivost,
- teško je odabrati težinsku funkciju koja daje zadovoljavajuća svojstva fluida i računalne efikasnosti te
- prikaz je grublji u odnosu na metode koje koriste mreže.

Praksa je pokazala da brzina ove metode je od velike važnosti kada želimo dobiti simulaciju fluida u realnom vremenu kao na primjer u video igrama, a nedavna istraživanja ove metode unaprjeđuju njenu preciznost i efikasnost.

2.2 Metoda Marker and Cell grid (MAC grid)

Ova metoda se temelji na diskretizaciji prostora u kojem se mjere parametri fluida. Diskretizacija je izvedena tako da se prostor podijeli u manje kvadrate odnosno kocke ovisno o dimenziji prostora - 2D ili 3D respektivno. Svaki takav djelić prostora, ćelija (eng. *cell*), sadrži informacije o uzorkovanom prostoru (poput tlaka, temperature itd.) te oznaku (eng. *marker*) nalazi li se na tom mjestu fluid. Takva podjela rezultira poljem ili rešetkom po kojoj ova metoda nosi i ime.

Osnovna varijanta ove metode pohranjuje podatke uzorkovane u centru svake ćelije, no Harlow i Welch su popularizirali varijantu u kojoj se komponente brzine pohranjuju na rubovima svake ćelije [1]. Vizualna reprezentacija tako određene ćelije prikazana je na slici 2.

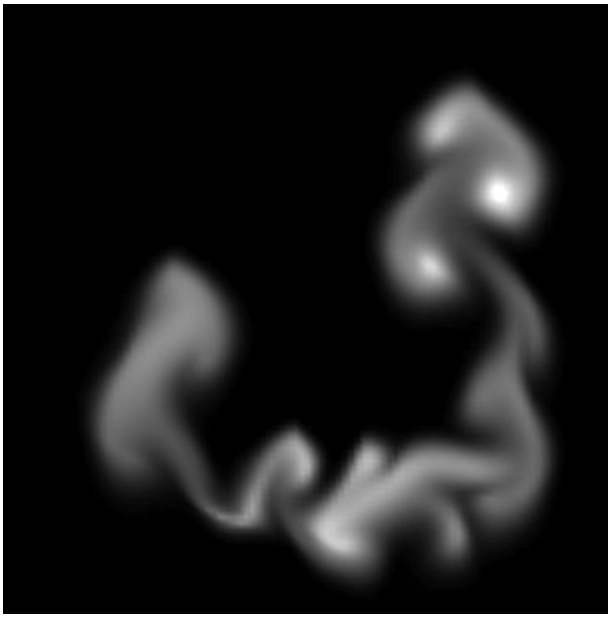


Slika 2. Primjer ćelije sa komponentama brzine na stranicama

Takva varijanta daje stabilniju i precizniju simulaciju fluida ali zahtjeva dodatnu interpolaciju u određivanju vektora brzine.

Izračun podataka sadržanih u ovako diskretiziranom prostoru svodi se na rješavanje Navier-Stokesovih jednadžbi. Za razliku od metode čestica koja koristi Navier-Stokesove za određivanje djelovanja među česticama u ovoj metodi se te jednadžbe razrješavaju za cijeli diskretizirani prostor odjednom što daje precizniju simulaciju nauštrb računalnih performansi.

Određivanje granica fluida je dosta jednostavno s obzirom da svaka ćelija sadrži oznaku nalazi li se u njoj fluid. Potrebno je samo još odrediti kojom bojom ćemo iscrtati fluid. Takav prikaz može rezultirati grubim prijelazima između prostora koji okupira fluid i praznog prostora pa se u ćelije dodaje vrijednost gustoće fluida. Potom se ona koristi za dodavanje transparentnosti rjeđem dijelu fluida što rezultira blagim prijelazom u prazan prostor. Primjer tako prikazanog fluida nalazi se na slici 3.



Slika 3. Simulacija fluida MAC grid metodom

Glavne prednosti ove metode su:

- preciznija simulacija dinamike fluida,
- lako očuvanje nestlačivosti fluida,
- brzina izvođenja se može unaprijediti korištenjem paralelizma i rada na grafičkim procesorima te
- prirodniji izgled miješanja više fluida i prijelaza u prazan prostor.

Glavne mane su:

- pojava nestabilnosti ako se uzme loš period uzorkovanja prostora i
- računalni zahtjevi su puno veći u odnosu na metodu sustava čestica.

Primjena ove metode se najčešće može vidjeti u znanstvenim radovima ili filmskoj industriji gdje je potrebno postići što stvarnije ponašanje i izgled fluida. Metoda je također lako proširiva na 3D prostor, no potrebni su veliki računalni resursi te visoko precizno simuliranje može trajati satima ako ne i danima.

3. Prilagodba MAC metode za simulaciju vatre

Prethodno je spomenuto da je vatra poseban slučaj fluida te je potrebno napraviti neke modifikacije postojećih metoda za dinamiku fluida. Kako me zanima potencijalni daljnji rad na modeliranju i animaciji vatre trebala mi je metoda koja se relativno lako može proširiti na 3D prostor te implementirati na grafičkim procesorima odlučio sam prilagoditi MAC metodu.

3.1 Navier-Stokesove jednadžbe

Kao što je već više puta spomenuto, opis dinamike fluida dan je Navier-Stokesovim jednadžbama:

$$\frac{\partial \boldsymbol{u}}{\partial t} = -(\nabla \cdot \boldsymbol{u})\,\boldsymbol{u} - \frac{1}{\rho}\nabla p + v\nabla^2 \boldsymbol{u} + \boldsymbol{F} \tag{1}$$

$$\nabla \cdot \boldsymbol{u} = 0 \tag{2}$$

Jednadžba (1) je generalna Navier-Stokesova jednadžba koja opisuje promjenu vektora brzine $\boldsymbol{u}$ kroz vrijeme t. Ta promjena ovisi o:

- očuvanju momenta opisanog prvim članom,
- gustoći ρ i tlaku p,
- viskoznosti v te
- vanjskim silama $\boldsymbol{F}$.

Druga jednadžba (2) opisuje da količina fluida koja ulazi u neki volumen je jednaka količini koja izlazi, odnosno ova jednadžba je uvjet nestlačivosti fluida.

3.2 Osnovni algoritam

Algoritam ove metode je primjena jednadžbi na diskretiziran prostor. Stabilnost simulacije ovisi o periodu uzorkovanja Δt koji se koristi u izračunu a dan je sljedećom formulom:

$$\Delta t = k\frac{h}{|\boldsymbol{u}_{max}|} \tag{3}$$

Vrijednost h je širina ćelije, k je konstanta koju zadaje korisnik kako bi skalirao vrijednost po želji, te $\boldsymbol{u}_{max}$ je najveća trenutna vrijednost brzine u prostoru.

Sljedeći korak je promjena brzina u prostoru kako je dano glavnom jednadžbom (1). Svaki član se posebno računa te zbraja neovisno o redoslijedu. Jedini član koji ovisi o redoslijedu izračuna je dodavanje tlaka. Taj član se računa na kraju jer istovremeno moramo zadovoljiti uvjet nestlačivosti (2).

Posljednji korak je pomicanje svih podataka u ćeliji koristeći novo izračunate vrijednosti brzina. Ovaj korak rezultira pomicanjem fluida kroz prostor praćenjem vektora brzina.

Pregled algoritma dan je na sljedećoj slici:

```
1. Izračunaj vremenski period

2. Pomakni polje brzina

   2.1.   Dodaj vanjske sile

   2.2.   Dodaj očuvanje momenta,
          tj. konvekciju (eng. convection)

   2.3.   Dodaj viskoznost

   2.4.   Izračunaj tlak uz uvjet nestlačivosti

   2.5.   Dodaj izračunati tlak

3. Pomakni fluid kroz polje brzina za vremenski period
```

Slika 4. Algoritam MAC metode

Detaljnije o samom načinu izračuna pojedinih članova bit će dano u poglavlju koje opisuje implementaciju.

3.3 Dodatne modifikacije

Kako bi modelirali vatru potrebno je dodati još neke parametre u ćelije te fizikalne efekte na osnovnu jednadžbu dinamike fluida (1).

Prvo se dodaje temperatura u centar ćelije kao novi podatak o fluidu. Sada pratimo gustoću i temperaturu fluida no kod gorenja ti parametri vremenom opadaju. Modeliranje disipacije temperature i gustoće dano je sljedećom jednadžbom:

$$x_{novi} = max(0, x_{trenutni} - k\,\Delta t) \tag{4}$$

x predstavlja temperaturu ili gustoću koja se smanjuje s korisnički zadanim koeficijentom k pomnoženim za period uzorkovanja Δt .

Sljedeći zahtjev je modeliranje gibanja toplog i hladnog zraka. Fedkiw, Stam i Jensen su takav efekt opisali jednostavnim izračunom sile "uzgona" sljedećom formulom [2].

$$\boldsymbol{F}_{uzgon} = \left(k_{rast}\,(q - q_{atm}) + k_{pad}\,\rho\right)\frac{\boldsymbol{g}}{\|\boldsymbol{g}\|} \tag{5}$$

Korisnik zadaje koeficijente uzdizanja toplog zraka k_{rast} i padanja hladnog zraka k_{pad} . Temperatura fluida je q a atmosfere q_{atm} . Gustoća fluida je ρ te se na kraju sve pomnoži s jediničnim vektorom gravitacije $\boldsymbol{g}$.

Posljednji bitan element vatre je vrtložnost (eng. *vorticity*). Potrebno je uvesti još jedan podatak u ćelije, rotaciju vektorskog polja brzina. Ona se računa prema izrazu:

$$n = \|\nabla \times \boldsymbol{u}\| \tag{6}$$

Gdje je n magnituda rotacije koju pohranjujemo, a je $\boldsymbol{u}$ vektor brzine. Dobivena vrijednost se koristi kako bi izračunali silu vrtložnosti $\boldsymbol{F}_{vort}$ prema formuli:

$$\boldsymbol{F}_{vort} = k_{vort}\left(\frac{\nabla n}{\|\nabla n\|} \times (\nabla \times \boldsymbol{u})\right) \tag{7}$$

Korisniku je ponovo omogućeno zadavanje koeficijenta k_{vort} kojim može kontrolirati jačinu vrtložnosti.

Naravno, postoje još mnogi efekti kod modeliranja vatre poput brzine protjecanja goriva, stvaranja dima i mogućnosti zapaljenja objekata. Pregled tih efekata dali su Nguyen, Fedkiw i Jensen [3].

4. Implementacija MAC metode

Implementacija je izvedena korištenjem programskog jezika C++ te su za grafički prikaz korištene biblioteke OpenGL i GLUT. Sav računski dio je izveden na CPU a ne GPU te nije korištena višedretvenost. Kod je napisan korištenjem uređivača teksta vim te preveden jezičnim procesorom gcc.

Program se sastoji od više dijelova i razreda a ovdje će biti prikazan dio vezan uz dinamiku fluida korištenjem MAC metode. Radi bolje preglednosti iz primjera koda uklonjeni su komentari ili neki nevažni dijelovi.

4.1 Algoritam

Rješavanje Navier-Stokesovih jednadžbi primjenom algoritma sa slike 4 implementirano je u razredu `SimpleFluidSolver` koji nasljeđuje apstraktno sučelje `IFluidSolver`. Sučelje deklarira dvije metode:

- `solveDensity` kojom se izvodi pomicanje fluida kroz polje vektora brzina (odnosno 3. korak algoritma) te
- `solveVelocity` u kojem se izračuna promjena brzina prema Navier-Stokesovim jednadžbama (2. korak algoritma).

```
void SimpleFluidSolver::solveVelocity(int n, SquareMatrix<float> &horVel,
        SquareMatrix<float> &vertVel, SquareMatrix<float> &horVelPrev,
        SquareMatrix<float> &vertVelPrev, float visc, float dt)
{
    addSources(n, horVel, horVelPrev, dt);
    addSources(n, vertVel, vertVelPrev, dt);
    horVel.swap(horVelPrev);
    vertVel.swap(vertVelPrev);
    diffuse(n, horVel, horVelPrev, visc, dt);
    diffuse(n, vertVel, vertVelPrev, visc, dt);
    project(n, horVel, vertVel, horVelPrev, vertVelPrev);
    horVel.swap(horVelPrev);
    vertVel.swap(vertVelPrev);
    advect(n, horVel, horVelPrev, horVelPrev, vertVelPrev, dt);
    advect(n, vertVel, vertVelPrev, horVelPrev, vertVelPrev, dt);
    project(n, horVel, vertVel, horVelPrev, vertVelPrev);
}
```

Metoda `solveVelocity` prima dimenzije polja n, podatke o brzinama kroz pomoćne razrede `SquareMatrix` te koeficijent viskoznosti `visc` i vremenski interval `dt`. Prvo se rješava član dodavanja vanjskih sila metodom

`addSources` izvedeno običnim matričnim zbrajanjem trenutnog stanja s prethodno podešenim silama te skaliranjem vremenskim intervalom.

Zatim se nad brzinama zove metoda `swap` kojima mijenjamo trenutno i prethodno stanje što je bitno za daljnje korake.

Metoda `diffuse` računa član viskoznosti:

$$v\nabla^2\boldsymbol{u} \tag{8}$$

Gdje se $\nabla^2\boldsymbol{u}$ računa tako da ∇^2 primijenimo na svaku komponentu vektora brzine koristeći formulu:

$$\begin{aligned} v\nabla^2 u(x,y) &= u(x+1,y) + u(x-1,y) + \\ &\quad u(x,y+1) + u(x,y-1) - 4u(x,y) \end{aligned} \tag{9}$$

Član konvekcije računa metoda `advect` primjenom metode praćenja čestice unazad (eng. *backwards particle trace*). Ideja je da iz jedne točke pratimo pomak brzine unazad za dani vremenski interval te odredimo novu točku. Zatim vrijednosti pohranjene u dobivenoj točci se spremaju u polaznu točku. Najjednostavnija implementacija ovog postupka zove se Eulerov korak (eng. *Euler step*) dana formulom:

$$\boldsymbol{y} = \boldsymbol{x} - \Delta t\, \boldsymbol{u}(\boldsymbol{x}) \tag{10}$$

Gdje je $\boldsymbol{x}$ koordinate trenutne točke a $\boldsymbol{y}$ nove. Kako nove koordinate ne moraju nužno biti točno unutar ćelije, zapisana vrijednost će biti rezultat bilinearne interpolacije među ćelijama oko dobivenih koordinata. Kod ove metode je bitno da imamo odvojene podatke iz prethodnog i trenutnog koraka za pravilan izračun što objašnjava prenošenje tih podataka u funkciju i korištenja metode `swap` u prethodnom koraku.

Računski najteži korak se nalazi u metodi `project` gdje se računa doprinos tlaka. Kako bi se odredio tlak potrebno je riješiti matričnu jednadžbu:

$$\boldsymbol{A}\,\boldsymbol{P} = \boldsymbol{B} \tag{11}$$

Tlak koji tražimo reprezentira matrica $\boldsymbol{P}$, matrica $\boldsymbol{A}$ su koeficijenti dobiveni negativnim brojem susjeda ćelije koji nisu prepreka fluidu (u ovoj

implementaciji nema prepreka pa svaka ćelija ima 4 takva susjeda što daje koeficijent -4). Vektor $\boldsymbol{B}$ se popunjava korištenjem formule:

$$\boldsymbol{b}_i = \frac{\rho\, h}{\Delta t}\,(\nabla \cdot \boldsymbol{u}_i) \tag{12}$$

Gdje je ρ gustoća fluida, h širina ćelije, Δt vremenski interval, a izraz $\nabla \cdot \boldsymbol{u}_i$ se računa formulom:

$$\nabla \cdot \boldsymbol{u}(x,y) = u_x(x+1,y) - u_x(x,y) + u_y(x,y+1) - u_y(x,y) \tag{13}$$

Tako dobiveni tlak se pribroji polju brzina prema izrazu:

$$\boldsymbol{u}_{novi}(x,y) = \boldsymbol{u}(x,y) - \frac{\Delta t}{\rho\, h}\,\nabla p(x,y) \tag{14}$$

A gradijent tlaka se računa formulom:

$$\nabla p(x,y) = \big(p(x,y) - p(x-1,y), p(x,y) - p(x,y-1)\big) \tag{15}$$

4.2 MAC polje

Diskretizirani prostor u kojem se izvodi simulacija vatre dan je sljedećim razredom:

```
class FireGrid: public FluidGrid2D {
    public:
        FireGrid(int n, float d, float r, float f,
                 float v, IFluidSolver *ifs);

        virtual void performStep(float dt);

    private:
        float diss;
        float rise;
        float fall;
        float vort;
        IFluidSolver *solver;

        SquareMatrix<float> horVel;
        SquareMatrix<float> horVelPrev;
        SquareMatrix<float> vertVel;
        SquareMatrix<float> vertVelPrev;
        SquareMatrix<float> density;
        SquareMatrix<float> densityPrev;
        SquareMatrix<float> temp;
        SquareMatrix<float> tempPrev;
        SquareMatrix<float> curl;
```

Razred `FireGrid` nasljeđuje osnovni razred `FluidGrid2D` koji predstavlja dvodimenzionalno MAC polje. Konstruktor prima sljedeće argumente:

- dimenzije MAC polja - `n`,
- konstantu disipacije topline i gustoće koja se pohrani u privatnu varijablu `diss`,
- konstantu podizanja toplog zraka i pada hladnog zraka koje se pohrane u privatne varijable `rise` i `fall` respektivno,
- konstantu vrtložnosti koja se pohranjuje u vort te
- pokazivač na razred koji implementira sučelje `IFluidSolver` za rješavanje jednadžbi, pohranjen u `solver`.

Razred sadrži još niz privatnih varijabli kojima prati podatke o fluidu i prostoru. Za pohranu tih podataka koristi se pomoćni razred `SquareMatrix` koji reprezentira kvadratnu matricu, odnosno 2D polje. Podaci koji se pohranjuju su brzine, gdje se brzina rastavlja na dvije komponente a to su horizontalna (`horVel`) i vertikalna (`vertVel`), gustoća (`density`), temperatura (`temp`) i magnituda rotacije (`curl`). Također, za svaki podatak se pamti i vrijednost u prethodnom koraku u varijabla sa sufiksom `Prev`.

Metoda `performStep` prima vremenski period `dt` te izvodi korak algoritma, a konkretna implementacija je pokazana u nastavku.

```
void FireGrid::performStep(float dt)
{
    vorticity();
    bouyancy();
    solver->solveVelocity(getDim(), horVel, vertVel,
                          horVelPrev, vertVelPrev, 0, dt);
    solver->solveDensity(getDim(), density, densityPrev,
                          horVel, vertVel, 0, dt);
    solver->solveDensity(getDim(), temp, tempPrev,
                          horVel, vertVel, 0, dt);
    dissipate(dt);
}
```

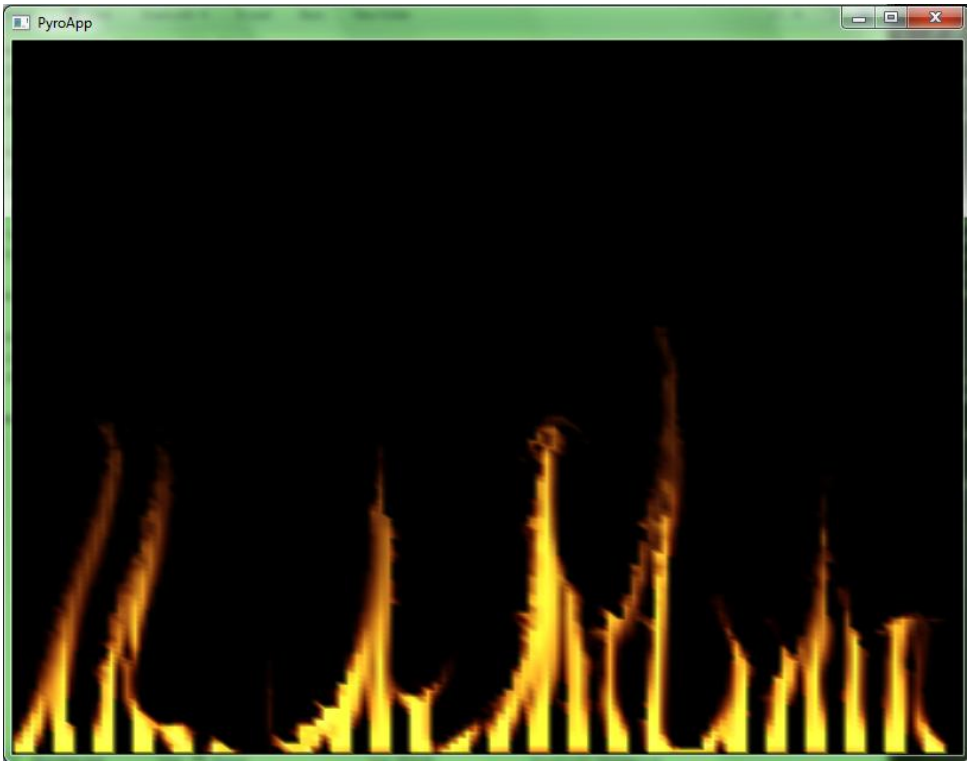
Prvo se postave vrijednosti sila vrtložnosti pozivom metode `vorticity` korištenjem prije spomenutih formula (6) i (7). Sile "uzgona" implementirane su u metodi `bouyancy` prema formuli (5). Zatim se koristi `solver` kako bi

primijenio algoritam sa slike 4 koristeći metode `solveVelocity` (2. korak algoritma) i `solveDensity` (3. korak algoritma) kojima se pomiče gustoća fluida i temperatura. Na kraju se umanji gustoća i temperatura metodom `dissipate` opisano formulom (4).

4.3 Problemi u razvoju

Tokom razvoja programa pojavilo se nekoliko problema vezanih uz izgled simulacije. Prvi i najvažniji je bio prikaz vatre gdje se jednostavna pretvorba gustoće u boju ne može koristiti kao kod bilo kojeg drugog fluida. Kada gledamo vatru ona je u sredini zlatno žuta a rubovi plamena prelaze u narančastu i tamnu crvenu boju. Kako bih uvjerljivo prikazao vatru, logičan izbor mi se činilo temeljiti boje na temperaturi vatre no prijenos temperature u vidljivi spektar nije jednostavna linearna funkcija. Nguyen, Fedkiw i Jensen su u svom radu spomenuli pretvorbu temperature u boju preko formula zračenja crnog tijela ali nažalost taj postupak nije opisan, već je dan kao ideja [3]. Rješenje ovog problema nisam mogao naći u literaturi pa sam morao pribjeći eksperimentiranju. Počeo sam jednostavnim množenjem RGB komponenata nekim faktorom i vrijednošću temperature fluida. Prikaz sam temeljio na slici:

Slika 5. Osnova bojanja simulirane vatre



Slika 6. Ostvaren prikaz vatre u programu

Nakon nekoliko iteracija postavljanja različitih faktora i uvjeta na temperaturu došao sam do zadovoljavajućeg izgleda prikazanog na slici 6 a konačna pretvorba je prikazana sljedećim kodom:

```
void FluidDisplay::setColor(float d, float t)
{
        float r = 1.0f;
        float g = t/50 * 0.9f;
        float b = t/50 * 0.2f;
        if (t > 100.0f) {
            r = 1.0f;
            g = 0.9f;
            b = 0.2f;
        }
        if (d < 0.1f) {
            r = g = b = 0.0f;
        }
        glColor4f(r, g, b, d / 20);
}
```

Funkcija `setColor` prima dva parametra: gustoću i temperaturu fluida u nekoj ćeliji. Prvo se postave osnovne vrijednosti RGB komponenti kada je temperatura manja od 100K, gdje padom temperature dominira crvena komponenta. Kod većih temperatura, boja se postavlja na žutu. Provjera da li

je gustoća manja od 0.1 služi kao granica reprezentacije fluida, odnosno gustoća manja od te vrijednosti se tretira kao da fluid više ne postoji. Konačna boja se prenosi u funkciju glColor4f gdje se predaje i alfa kanal boje (prozirnost) određena gustoćom čime se postiže ublaženo nestajanje fluida.

Manji problemi su se pojavili s parametrima simulacije. Na primjer, ako se stavi prevelika vrijednost koeficijenta za pad hladnog zraka, vatra počinje gorjeti prema dole. Povećanje brzine uzdizanja toplog zraka uzrokuje pojavu visokog plamena što vizualno nije poželjno a može i destabilizirati simulaciju pojavom velikih brzina. Rješenje je ponovno ležalo u testiranju različitih vrijednosti parametara i njihovom usklađivanju na temelju promatranog.

4.4 Performanse

Testiranje programa izvedeno je na četvero jezgrenom procesoru AMD Athlon II X4 630, radnog takta 2.8 GHz te radnom memorijom DDR3 od 4 GB na 64-bitnom operacijskom sustavu Windows 7. Kao mjera performansi programa korištene su sličice u sekundi (FPS, eng. *Frames per Second*) kako ih mjeri alat FRAPS. Za potrebe ovog rada korištena je *trial* verzija tog programa. Rezultati su prikazani u sljedećoj tablici:

Tablica 1. Ovisnost dimenzija o brzini izvođenja

Dimenzije MAC polja	FRAPS / FPS
80 x 80	63
100 x 100	40
120 x 120	29
150 x 150	18

4.5 Buduća unaprjeđenja

Najzanimljivije unaprjeđenje ovog simulatora je proširenje na treću dimenziju. Odabrana metoda je za to lako proširiva ali bi simulacija zahtijevala puno više računalnih resursa. Kako bi se unaprijedile performanse poželjno bi bilo napraviti implementaciju na grafičkom procesoru. Takva izvedba bi ponudila paralelizam koji ne bi trebalo posebno

prilagođavati za metodu MAC polja. Dodatno ubrzanje bi ponudile i bolje strukture podataka. Trenutno se cijelo polje inicijalizira u memoriji no velika većina ćelija je prazna, bolje rješenje bi bilo da je polje reprezentirano rijetkom matricom korištenjem mape raspršenog adresiranja.

Ovako poboljšana implementacija programa bi već na današnjim računalima mogla prikazivati uvjerljivu 3D simulaciju vatre u realnom vremenu.

5. Zaključak

Dinamika fluida je prilično komplicirano područje fizike za računalnu simulaciju. Osim što je odabrana metoda zahtjevna za računalne resurse, razumijevanje jednadžbi se činilo jednako zahtjevno. Kroz traženje materijala naišao sam na članak koji objašnjava svaki član jednadžbe te kako se on implementira [4]. Stečenim znanjem u toku izrade ovog rada zaključujem da je primjena MAC metode za fizički temeljeno modeliranje vatre relativno jednostavno te daje vrlo dobre rezultate. Metoda je prilagodljiva paralelizmu, a današnji fokus razvitka računalne moći leži upravo na tom području što će umanjiti njene mane. Primjene računalne dinamike fluida su raširene na mnoga područja te se nastavlja rad na unaprjeđenjima postojećih algoritama. Metoda MAC polja trenutno nudi najveću razinu detaljnosti simulacije te napretkom tehnologije bi mogla postati popularan izbor modeliranja interaktivnih fluida u realnom vremenu.

6. 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. 2182-2189
2. Fedkiw R., Stam J., Jensen H. W. Visual simulation of smoke. Proceedings of the 29th annual conference on Computer Graphics and interactive techniques. ACM Press. 2001. 15-22
3. Nguyen D., Fedkiw R., Jensen H. W. Physically Based Modeling and Animation of Fire. SIGGRAPH 2002. ACM TOG 21. 721-728
4. Cline D., Cardon D., Egbert P. K. *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.

7. Sažetak

Naslov: Animacija fizikalno temeljenog modela vatre

Ovaj rad opisuje računalnu simulaciju vatre. Kako je model vatre poseban slučaj fluida, predstavljene su često korištene metode u računalnoj dinamici fluida. Glavni cilj rada bila je izrada programa koji demonstrira kako se jedna od prethodno spomenutih metoda može iskoristiti za animaciju fizikalno temeljenog modela vatre. Opisana je implementacija i prilagodba metode *MAC* polja. Aplikacija je izrađena u programskom jeziku C++ te je korištena biblioteka OpenGL za prikaz i animaciju vatre. Rad je zaključen pregledom prednosti i mana odabrane metode te njenih mogućih unaprjeđenja i modifikacija.

Ključne riječi: vatra, fizikalna animacija, fluid, računalna dinamika fluida, jednadžbe Navier-Stokes, *MAC* polje

8. Abstract

Title: Physically based models of fire

This thesis describes computational simulation of fire. Since the fire model is a special case of a fluid, common methods in computational fluid dynamics are presented. The main goal of the thesis was the creation of a program with which to demonstrate how one of the aforementioned methods can be applied for physically based modelling and animation of fire. Implementation and adaptation of the *MAC* grid method is described. The application was developed with C++ programming language and uses OpenGL *API* for rendering and animating the fire. The thesis is concluded with the overview of advantages and disadvantages of the chosen method as well as potential improvements and modifications.

Keywords: fire, physically based animation, computational fluid dynamics, Navier-Stokes equations, *MAC* grid

9. Privitak

9.1 Upute

Program se pokreće pokretanjem izvršne datoteke pod nazivom `pyro`. Dodatne opcije programu se mogu predati pokretanjem iz konzole i dodavanjem sljedećih parametara ovim redom:

- dimenzije rešetke
- koeficijent disipacije,
- koeficijent brzine podizanja toplog zraka,
- koeficijent brzine padanja hladnog zraka te
- koeficijent vrtložnosti vatre.

Nakon što se program pokrene, pojavi se prozor u kojem je ostvaren 2D prikaz vatre.

Interakcija sa simulacijom omogućena je uporabom miša, kojim korisnik može dodavati vjetar u simulaciju. Dodavanje je ostvareno držanjem lijeve tipke miša i povlačenjem po ekranu. Ovisno o brzini i smjeru povlačenja miša ovisi brzina i smjer stvorenog vjetra.

Dimenzije prozora aplikacije se mogu mijenjati pritiskom lijeve tipke miša na rub prozora i povlačenjem kako je standardno u grafičkim sučeljima temeljenih na prozorima. Povećanjem prozora će se bolje uočiti diskretiziranost prostora pa za bolji vizualni doživljaj preporučeno je ostaviti osnovne postavke ili smanjiti prozor.

Zatvaranje programa ostvaruje se lijevim klikom na znak X odnosno ekvivalentnim simbolom u grafičkom sučelju na rubu prozora programa ili pritiskom slova *q* na tipkovnici.