SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 542

VIZUALIZACIJA DINAMIKE FLUIDA METODOM HIDRODINAMIKE ZAGLAĐUJUĆIH ČESTICA

Hrvoje Hemen

Zagreb, lipanj, 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 542

Pristupnik:	**Hrvoje Hemen (0036523139)**
Studij:	Računarstvo
Profil:	Programsko inženjerstvo i informacijski sustavi
Mentor:	prof. dr. sc. Krešimir Trontl

Zadatak: **Vizualizacija dinamike fluida metodom hidrodinamike zaglađujućih čestica**

Opis zadatka:

Računalne simulacije pretvaraju složeni fizikalni model definiran u kontinuiranoj domeni u diskretan oblik koji je moguće riješiti uporabom računala. Vizualizacija simulacije omogućava korisniku bolju percepciju i razumijevanje fizikalne pojave opisane simulacijom. Stoga je cilj ovog rada razvoj aplikacije za vizualizaciju dinamike fluida metodom hidrodinamike zaglađujućih čestica - SPH. U radu je potrebno analizirati fizikalne osnove dinamike fluida kao i osnovne karakteristike SPH metode. Upotrebom programskog jezika C# potrebno je vizualizirati ponašanje fluida u različitim realnim uvjetima.

Rok za predaju rada: 28. lipnja 2024.

Želim se zahvaliti mentoru Krešimiru Trontlu-

Sadržaj

1. Uvod

1.1. Cilj rada

Cilj ovog rada bio je napraviti realnu simulaciju dinamike fluida. Korištena metoda bila je metoda hidrodinamike zaglađujućih čestica (SPH). Inspiracija za ovaj rad bio je jedan YouTube video Sebastiana Laguea koji govori o simulaciji vode u Unityju.

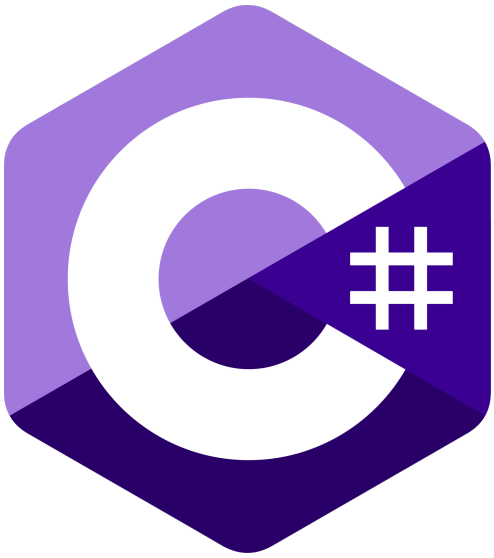
1.2. Ukratko o radu

U sklopu ovog rada obrađeno je sve potrebno za samostalnu izradu ovog rada uključujući i postavljanje razvojnog okruženja.

Rad je pisan u c# programskom jeziku u sklopu Unityja, te je za vizualizaciju korišten Unityjev dvodimenzionalni vizualizator.

2. Tehnologije

2.1. C#



Slika 2.1. C Sharp Logo [1]

C# ("C-sharp") je objektno orijentirani programski jezik visoke razine razvijen od strane Microsofta. Jezik je nastao ranih 2000-ih kao dio .NET inicijative, s ciljem da kombinira računalnu snagu C++ s jednostavnošću i sigurnošću Jave.

- **Razvoj i povijest:** C# je prvi put predstavljen 2000ġodine i brzo je postao popularan zbog svoje sličnosti s C i C++ jezicima, kao i zbog svoje integracije s .NET frameworkom. Jezik je dizajniran kako bi bio jednostavan za učenje, strogo tipiziran i prijenosan na različite operacijske sustave.

- **Sintaksa i struktura:** C# sintaksa je vrlo slična onoj u Javi, gdje svaka naredba završava sa točka-zarezom (;). Kod je organiziran u klase i metode, a dijelovi koda su omeđeni vitičastim zagradama ({}). Jedna od ključnih razlika između C# i Jave je

mogućnost preopterećenja osnovnih operacija u C#. Na primjer, možete definirati kako se operator + ponaša za prilagođene klase.

- **Prednosti i upotreba:** C# je vrlo moćan jezik s mnogim značajkama kao što su LINQ (Language Integrated Query), async/await za jednostavno upravljanje asinkronim operacijama, i podrška za razne paradigme programiranja uključujući objektno orijentirano, funkcionalno i imperativno programiranje.

- **Integracija s Visual Studio:** C# je potpuno integriran s Visual Studio, jednim od najmoćnijih razvojnih okruženja, što omogućava brzi razvoj, testiranje i implementaciju aplikacija.

- **Ekosustav i knjižice:** C# ima bogat ekosustav knjižica kao što su ASP.NET za web razvoj, Xamarin za mobilne aplikacije, te Unity za razvoj igara.

2.2. Unity

Slika 2.2. Unity Logo [2]

Unity je razvojno okruženje i pogonski sklop za igre koje omogućava razvoj 2D, 2.5D i 3D igara. Od svog nastanka 2005ġodine, Unity se kontinuirano razvija i postaje jedan od najpopularnijih alata za razvoj igara na tržištu.

- **Razvoj i povijest:** Unity je lansiran 2005. godine s ciljem da olakša razvoj igara, pružajući pristupačne alate i dokumentaciju za programere svih razina vještina. Od tada, Unity je narastao i zauzeo značajan udio na tržištu razvoja igara.
- **Platforme i podrška:** Jedna od najvećih prednosti Unityja je njegova sposobnost razvoja za različite platforme, uključujući mobilne uređaje, desktop, web, konzole i virtualnu stvarnost. To omogućava programerima da razvijaju igre koje mogu biti lako prenesene i distribuirane na više platformi.
- **Licenciranje i pristupačnost:** Unity nudi nekoliko razina licenci, uključujući besplatnu verziju koja omogućava novim programerima ulazak u svijet razvoja igara. Besplatna verzija je opsežna i omogućava programerima da istraže sve značajke Unityja bez početnih troškova.
- **Dokumentacija i zajednica:** Unity je poznat po svojoj opsežnoj dokumentaciji i aktivnoj zajednici. Mnogi resursi, kao što su forumi, vodiči, tečajevi i online podrška, dostupni su kako bi pomogli programerima da brzo nauče i rješavaju probleme.
- **Alati i integracija:** Unity nudi širok spektar alata za razvoj, uključujući svoj IDE, alate za animaciju, nativni fizički simulator, te podršku za nadogradnje razvojene

od strane drugih ljudi.

- **Renderiranje i grafika:** Unity koristi napredne tehnike renderiranja i grafičke mogućnosti, uključujući podršku za High Definition Render Pipeline (HDRP) i Universal Render Pipeline (URP), što omogućava visoku kvalitetu vizualnih efekata i performansi na raznim uređajima.

3. Teorijska podloga

3.1. Pristupi računalnoj simulaciji fluida

Kada govorimo o simulaciji fluida, postoji nekoliko različitih pristupa koji se koriste u računalnim znanostima i inženjerstvu. Ovaj rad se fokusira na SPH (Smoothed Particle Hydrodynamics) metodu, no važno je razumjeti širi spektar tehnika koje su dostupne i njihove primjene. U ovom poglavlju, raspravit ćemo o najvažnijim metodama simulacije fluida, njihovim karakteristikama, prednostima i nedostacima.

3.1.1. Simulacije bazirane na česticama

Simulacije bazirane na česticama su jedne od najintuitivnijih i najčešće korištenih metoda za simulaciju fluida. U ovim metodama, svaka čestica predstavlja mali dio fluida, kao što je kapljica vode, te nosi svojstva kao što su masa, brzina i vektor smjera kretanja. Ove simulacije omogućavaju vrlo detaljan i dinamičan prikaz tekućina.

- **Osnovni princip:** Svaka čestica ima svoje fizičke karakteristike koje se koriste za izračunavanje među-čestičnih sila poput tlaka, gustoće i viskoznosti. Na temelju tih sila, simulacija izračunava nova svojstva čestica za svaki korak simulacije. Ovaj proces se ponavlja iterativno, omogućujući simulaciji da prikaže dinamiku fluida kroz vrijeme.

- **SPH (Smoothed Particle Hydrodynamics):** SPH je najpoznatija metoda simulacije fluida bazirana na česticama. U SPH metodi, interakcije između čestica se izračunavaju pomoću glatkih interpolacijskih funkcija koje omogućuju realističnu simulaciju fluida. SPH će biti detaljno obrađena u zasebnom poglavlju ovog rada.

- **DEM[3] (Discrete Element Method):** DEM metoda se koristi za simulaciju gra-

nularnih materijala poput piljevine, pijeska i šljunka. Ova metoda uzima u obzir međučestične sile poput trenja i elastičnosti, te stavlja velik naglasak na primjenu Newtonovih zakona. DEM se najčešće koristi u rudarskom inženjerstvu, ali također ima primjene u farmaceutskoj industriji, gdje se koristi za simulaciju procesa kao što je miješanje praha.

3.1.2. Simulacije bazirane na 2D polju

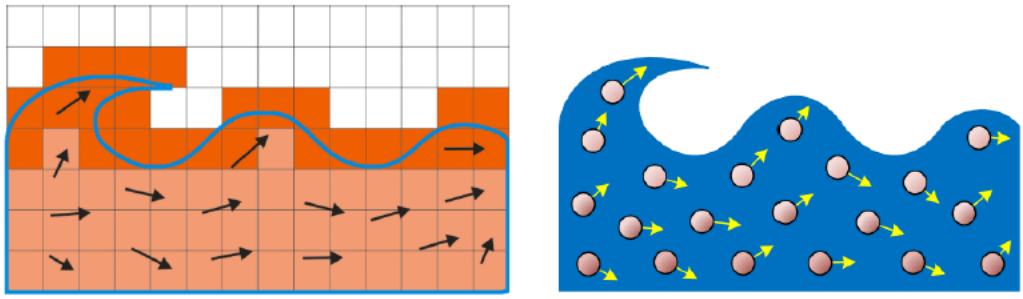
Simulacije bazirane na 2D polju koriste mrežu za reprezentaciju prostora fluida. Umjesto čestica, svojstva fluida se pohranjuju u ćelijama ove mreže, što omogućava drugačiji pristup simulaciji.

- **Osnovni princip:** Prostor simulacije se dijeli na ćelije u 2D mreži. Svaka ćelija pohranjuje informacije o fizikalnim svojstvima fluida kao što su brzina, tlak i gustoća. Simulacija se odvija iterativno, pri čemu se fizikalni zakoni primjenjuju na svaku ćeliju kako bi se izračunala nova stanja za svaki korak simulacije.

- **Primjena:** Ove simulacije se često koriste za modeliranje toka fluida u jednostavnim geometrijama i za simulacije gdje su granice jasno definirane. Primjeri uključuju simulaciju strujanja zraka preko krila aviona ili simulaciju morskih struja u ograničenim područjima.

- **Ograničenja:** Jedan od glavnih nedostataka simulacija baziranih na 2D polju je njihova računalna složenost. Kako se veličina mreže povećava, broj ćelija raste kvadratno, što značajno povećava potrebne računalne resurse. Zbog toga, ove metode nisu uvijek prikladne za simulacije velikih površina ili kompleksnih tokova.

- **Prednosti:** Unatoč ograničenjima, simulacije bazirane na 2D polju mogu pružiti vrlo precizne rezultate za specifične probleme. Mogu biti efikasnije od metoda baziranih na česticama za simulacije gdje su detalji pojedinačnih čestica manje važni od ukupnog toka fluida.



Slika 3.1. Grid based, Particle based simulation [4]

3.2. SPH metoda

3.2.1. Općenito o metodi

SPH[5] (Smoothed Particle Hydrodynamics) metoda je računska metoda koja se koristi za simulaciju fluida. Ova metoda je bazirana na česticama, pri čemu svaka čestica predstavlja mali volumen fluida. SPH metoda izračunava fizikalna svojstva fluida na temelju međučestičnih interakcija, omogućujući simulaciju fluida u složenim i nepravilnim prostorima bez potrebe za 2D poljem.

- **Osnovni princip:** U SPH metodi, prostor se ne dijeli na ćelije kao u metodama baziranim na mreži, već se koristi diskretna kolekcija čestica. Svaka čestica nosi informacije o fizikalnim svojstvima kao što su masa, brzina, gustoća i tlak. Interakcije između čestica izračunavaju se pomoću glatkih interpolacijskih funkcija, što omogućava kontinuirani prikaz fizikalnih veličina.

- **Prednosti:** Jedna od glavnih prednosti SPH metode je njena sposobnost da simulira fluide u složenim geometrijama i nepravilnim prostorima. Budući da ne koristi 2D polje, SPH metoda je fleksibilnija i može se lako prilagoditi različitim vrstama problema. Također, njena složenost u odnosu na broj čestica je manja nego kod metoda koje koriste 2D polje, posebno kada je gustoća čestica visoka.

- **Nedostaci:** Glavni nedostatak SPH metode je potreba za velikim brojem čestica kako bi se postigla visoka preciznost simulacije, što može povećati zahtjevnost simulacije i dodatno ju usporiti.

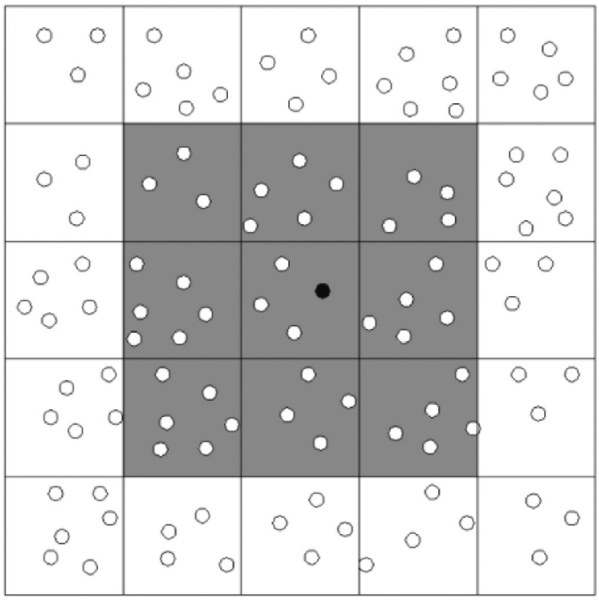
3.2.2. Koraci simulacije

Inicijalizacija simulacije

Na samom početku simulacije potrebno je definirati osnovne parametre koji će utjecati na ponašanje simulacije. Ovi parametri uključuju jačinu viskoznosti, koja određuje otpor tekućine prema promjeni oblika, te koeficijente međučestičnog odbijanja i privlačenja, koji utječu na način na koji se čestice međusobno odbijaju ili privlače. Osim toga, potrebno je definirati početne uvjete simulacije, kao što su početne pozicije i brzine čestica, kao i rubne uvjete koji definiraju granice simulacijskog prostora. Također, određuju se fizičke konstante kao što su gravitacija i koeficijenti pritiska, koji će biti korišteni u daljnjim koracima simulacije.

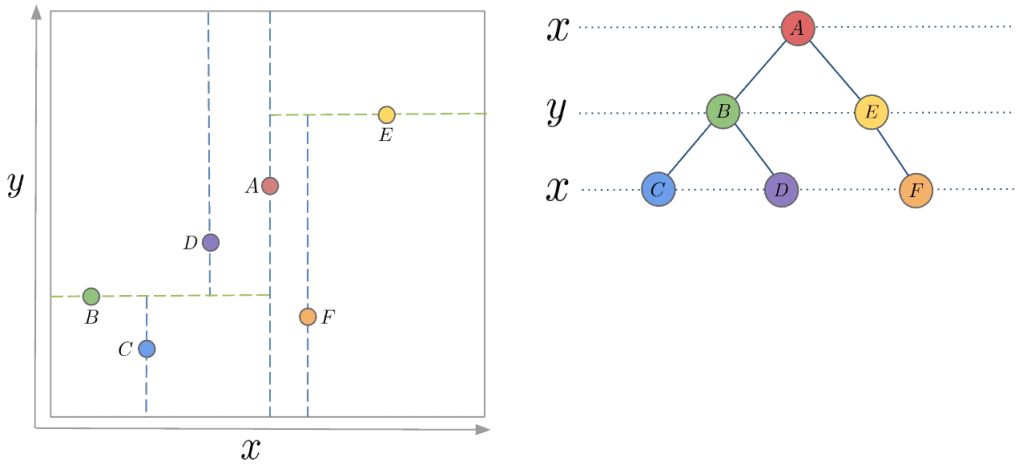
Traženje susjeda

Prvi "pravi" korak simulacije je traženje susjeda za svaku česticu. Susjedi su čestice koje se nalaze unutar određenog radijusa od ciljne čestice, i vrlo je važno znati njihove pozicije kako bi se na čestice mogle primijeniti sile koje ovise o udaljenosti, poput viskoznosti i pritiska. Najčešći pristup traženju susjeda je korištenje uniformne mreže ili 2D polja, gdje se svaka čestica na početku simulacije smješta u odgovarajuću ćeliju mreže. Kada se traže susjedi, provjeravaju se čestice u bliskim ćelijama mreže, što omogućava brzo i efikasno pretraživanje. Ova metoda značajno smanjuje broj potrebnih provjera udaljenosti između čestica, čime se optimizira cijeli proces simulacije. Uniformna mreža omogućava jednostavno indeksiranje i brzi pristup podacima, što je ključno za izvođenje simulacija u realnom vremenu.



Slika 3.2. Uniform grid searching method [6]

Drugi česti pristup traženju susjeda je korištenje stablastih struktura podataka, kao što su k-d stabla ili oktalna stabla. U ovom pristupu, čestice se organiziraju u hijerarhijsku strukturu stabla, gdje svaka čvor stabla predstavlja podskup čestica. Kada se traže susjedi za određenu česticu, pretraga se provodi spuštanjem niz stablo, čime se brzo identificiraju potencijalni susjedi. Ovaj pristup može biti vrlo učinkovit za velike skupove čestica jer smanjuje broj potrebnih provjera udaljenosti.



Slika 3.3. K-D stablo [7]

Najjednostavniji, ali i najmanje efikasan pristup traženju susjeda je brute-force metoda, gdje se za svaku česticu provjerava udaljenost od svake druge čestice. Ovaj pristup

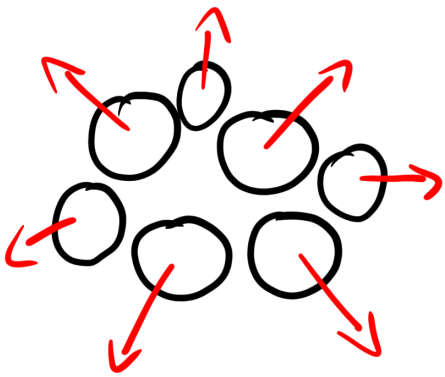
ima kvadratnu složenost, što znači da se broj provjera eksponencijalno povećava s brojem čestica. Zbog toga je ovaj pristup previše spor za praktičnu upotrebu u simulacijama s velikim brojem čestica.

Računanje i primjena međučestičnih sila

Nakon pronalaska susjeda slijedi ključni korak u kojem se računaju međučestične sile. Ovo je najvažniji korak jer upravo ove sile su one koje pokreću čestice da se gibaju poput vode. Čestice ne smiju biti unutar jedne druge, ali se svejedno moraju kretati zajedno, a skupina čestica mora se ponašati kao kohezivna jedinica. Najbitnije sile u ovoj fazi su sile pritiska i sile viskoznosti.

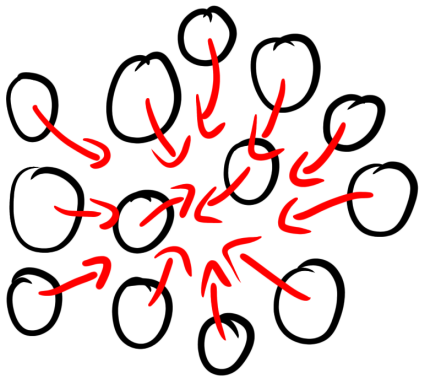
Prvo se računa gustoća, a zatim pomoću nje pritisak i onda nakon viskoznost.

Pritisak je bitan jer on tjera čestice jedne iz drugih kako nebi zapele jedna u drugu.



Slika 3.4. Sile pritiska

S druge strane, viskoznost je sila koja drži čestice na okupu. Viskoznost povlači čestice jedne prema drugima, omogućujući im da se kreću kao jedinstvena masa. Što je viskoznost jača, fluid koji simuliramo bit će "tvrđi" i više će nalikovati krutini, jer će unutarnje sile biti toliko jake da će se vanjske sile poput gravitacije moći zanemariti.



Slika 3.5. Sile viskoze

Nakon što su sile pritiska i viskoznosti izračunate, primjenjuju se na svaku česticu kako bi se dobila nova brzina i položaj. Konačne pozicije i brzine čestica se ažuriraju na temelju ovih sila, omogućujući simulaciji da napreduje kroz vrijeme.

Balansiranje ovih sila ključno je za postizanje realističnih simulacija fluida koje odgovaraju stvarnom ponašanju materijala. Kada su ove sile pravilno izračunate i primijenjene, simulacija će prikazivati prirodno gibanje tekućine, s česticama koje se pravilno međusobno odbijaju i privlače, oponašajući stvarno ponašanje vode ili drugih tekućina.

Upravljanje rubnim uvjetima i sudarima

Upravljanje rubnim uvjetima i sudarima predstavlja kritičan aspekt svake simulacije fluida, uključujući one koje koriste SPH metodu. Rubni uvjeti definiraju kako se fluid ponaša na granicama simulacijskog prostora, dok upravljanje sudarima osigurava realistične interakcije između fluida i čvrstih objekata ili granica.

- **Rubni uvjeti:** Postoji nekoliko vrsta rubnih uvjeta koji se mogu primijeniti u SPH simulacijama, uključujući:
  - *Dirichletovi rubni uvjeti:* Ovi uvjeti specificiraju vrijednosti funkcije (npr. brzine ili gustoće) na granicama simulacijskog prostora. Primjerice, u simulaciji

fluida, brzina fluida na zidovima posude može biti postavljena na nulu kako bi se simuliralo stanje mirovanja.

- *Neumannovi rubni uvjeti:* Ovi uvjeti specificiraju vrijednosti derivacija funkcije na granicama. U kontekstu simulacije fluida, ovo može značiti postavljanje gradijenta brzine ili tlaka na određene vrijednosti na granicama.

- *Periodični rubni uvjeti:* Ovi uvjeti omogućuju česticama koje napuste jednu stranu simulacijskog prostora da se ponovno pojave na suprotnoj strani, stvarajući efekt beskonačnog prostora.

Pravilno postavljanje rubnih uvjeta ključno je za održavanje stabilnosti i realizma simulacije. Primjerice, u simulacijama gdje fluid dodiruje čvrste zidove, potrebno je osigurati da fluid ne procuri ("clippa") kroz zidove i da se ponaša u skladu s očekivanim fizikalnim zakonima.

- **Upravljanje sudarima:** U simulacijama fluida, sudari mogu nastati između čestica fluida, kao i između čestica fluida i čvrstih objekata. Postoji nekoliko metoda za upravljanje sudarima:

  - *Phantom particles (fantomske čestice):* Ova metoda uključuje postavljanje dodatnih čestica duž granica čvrstih objekata. Fantomske čestice služe za simulaciju sile odbijanja, osiguravajući da čestice fluida ne prodiru kroz granice.

  - *Boundary handling (upravljanje granicama):* Ova metoda koristi posebne algoritme za detekciju i odgovor na sudare čestica fluida s granicama simulacijskog prostora. Na primjer, može se koristiti refleksija brzina kako bi se simulirao odbijanje fluida od zidova.

  - *Friction and adhesion (trenje i adhezija):* Ove sile igraju važnu ulogu u simulaciji interakcija između fluida i čvrstih površina. Trenje može smanjiti brzinu fluida pri kontaktu sa zidom, dok adhezija može uzrokovati da fluid "prianja" uz površine, što je važno za simulaciju efekata poput mokrih površina.

Upravljanje rubnim uvjetima i sudarima zahtijeva balansiranje između preciznosti i računalne efikasnosti. Pravilna implementacija ovih aspekata ključna je za stva-

ranje realističnih i stabilnih simulacija fluida koje se mogu koristiti u različitim aplikacijama, od računalne grafike do inženjerskih simulacija.

4. Programska implementacija

U ovom poglavlju detaljno će biti opisani programske koraci implementacije SPH simulacije fluida koristeći Unity i C#. Fokus će biti na osnovama Unity okruženja i Unity fizičkog simulatora, što će omogućiti razumijevanje integracije SPH metode u Unity okruženje.

4.1. Osnove Unity okruženja

Unity je moćan razvojni alat koji omogućava stvaranje igara i simulacija u 2D, 2.5D i 3D okruženjima. Njegova fleksibilnost i pristupačnost čine ga idealnim izborom za razvoj složenih fizikalnih simulacija kao što je SPH metoda za simulaciju fluida.

4.1.1. Unity Editor

Unity Editor je glavno razvojno okruženje koje omogućava korisnicima da dizajniraju, razvijaju i testiraju svoje aplikacije. Editor se sastoji od nekoliko ključnih komponenti:

- **Scene View:** Omogućava pregled i uređivanje scena igre. U ovom prikazu mogu se postavljati objekti, prilagođavati njihove pozicije, rotacije i veličine, te dodavati komponente.
- **Game View:** Omogućava pregled igre onako kako će je korisnik vidjeti kada je pokrene. Ovaj prikaz se koristi za testiranje i debugiranje.
- **Hierarchy Window:** Prikazuje sve objekte u sceni u hijerarhijskom prikazu. Ovdje se mogu organizirati objekti, stvarati roditeljsko-dječji odnose i upravljati strukturom scene.
- **Inspector Window:** Omogućava pregled i uređivanje svojstava odabranih obje-

kata. Ovdje se mogu dodavati komponente te mijenjati vrijednosti postojećih komponenti i skripti.

- **Project Window:** Prikazuje sve resurse projekta, uključujući skripte, modele, teksture i zvukove. Ovdje se može upravljati resursima vašeg projekta.

4.1.2. Komponente i GameObjecti

U Unityju, svaki objekt u sceni je GameObject. GameObjecti mogu imati različite komponente koje određuju njihovo ponašanje i izgled. Neke od ključnih komponenti uključuju:

- **Transform:** Svaki GameObject ima komponentu Transform koja pohranjuje informacije o poziciji, rotaciji i skali objekta.
- **Rigidbody:** Komponenta koja omogućava fizičku simulaciju objekta, uključujući gravitaciju, sile i sudare.
- **Collider:** Komponenta koja definira oblik objekta za detekciju sudara. Postoje različite vrste collidera, uključujući Box Collider, Sphere Collider i Mesh Collider, ovisno o objektu na koji stavljamo collider taj tip ćemo koristiti.
- **Script:** Komponente koje dodaju prilagođeno ponašanje objektima koristeći C# skripte.

4.1.3. Skripte u C#

Unity koristi C# kao glavni programski jezik za razvoj skripti. Skripte omogućavaju programerima da definiraju prilagođeno ponašanje za GameObjecte. Neke ključne značajke skripti u Unityju uključuju:

- **Monobehaviour:** Svi skriptni objekti koji žele biti povezani s GameObjectima trebaju naslijediti klasu Monobehaviour. Ova klasa omogućava pristup osnovnim metodama životnog ciklusa kao što su Start, Update, FixedUpdate i LateUpdate, koje se zovu u određeno vrijeme ciklusa.
- **Metode životnog ciklusa:** *Start* se poziva jednom kada se skripta prvi put po-

krene, *Update* se poziva jednom po taktu, *FixedUpdate* se poziva na fiksnim vremenskim intervalima i koristi se za fiziku, dok se *LateUpdate* poziva nakon svih Update metoda i koristi se za finaliziranje kretanja objekata.

- **Upravljanje komponentama:** Skripte mogu dodavati, uklanjati i mijenjati komponente GameObjecta, omogućavajući dinamičko ponašanje tijekom igre.

4.2. Osnove Unity fizičkog simulatora

Unity fizički simulator omogućava realističnu simulaciju fizikalnih interakcija među objektima. Ovaj simulator koristi komponente kao što su Rigidbody i Collider za upravljanje fizikalnim svojstvima objekata i detekciju sudara.

4.2.1. Rigidbody komponenta

Rigidbody komponenta dodaje fizikalna svojstva GameObjectu, omogućavajući mu da reagira na gravitaciju, sile i sudare. Ključne postavke Rigidbody komponente uključuju:

- **Mass:** Masa objekta koja utječe na to kako se objekt ponaša pod utjecajem sila.
- **Drag:** Otpornost na kretanje koja usporava objekt.
- **Angular Drag:** Otpornost na rotaciju.
- **Use Gravity:** Opcija koja određuje hoće li objekt biti pod utjecajem Unityjevog gravitacijskog simulatora.
- **Is Kinematic:** Ako je omogućeno, objekt neće biti pod utjecajem fizikalnih sila, već će se njime upravljati isključivo putem skripti.

4.2.2. Collider komponenta

Collider komponenta definira oblik objekta za detekciju sudara. Postoji nekoliko vrsta collidera koje ćemo koristiti ovisno o izgledu objekta ili slučaja korištenja u kojem se nalazimo:

- **Box Collider:** Pravokutni collider koji definira objekt kao kutiju, Kutija u nekoj igri.

- **Sphere Collider:** Kuglasti collider koji definira objekt kao sferu, Lopta u nogometnoj igri.
- **Capsule Collider:** Collider u obliku kapsule, Igrač u pucačkim igrama.
- **Mesh Collider:** Collider koji koristi mrežu (mesh) za definiranje složenih oblika.

Collideri omogućuju detekciju sudara i interakciju među objektima. Kada se dva collidera sudare, Unity fizički simulator izračunava reakcije temeljem fizikalnih svojstava objekata.

4.2.3. Simulacija fluida u Unityju

Za implementaciju SPH metode za simulaciju fluida u Unityju, potrebno je koristiti prilagođene skripte koje definiraju ponašanje čestica fluida. Ključni koraci uključuju:

- **Inicijalizacija čestica:** Postavljanje početnih pozicija, brzina i svojstava čestica.
- **Traženje susjeda:** Korištenje algoritama za brzo traženje susjeda unutar određenog radijusa.
- **Izračun međučestičnih sila:** Primjena pritiska, viskoznosti i drugih sila na temelju udaljenosti među česticama.
- **Ažuriranje položaja:** Korištenje numeričkih metoda za integraciju i ažuriranje položaja i brzina čestica kroz vrijeme.
- **Upravljanje sudarima:** Implementacija algoritama za detekciju i odgovor na sudare između čestica i s rubnim uvjetima.

Ove osnove Unity okruženja i fizičkog simulatora postavljaju temelje za razumijevanje kako implementirati kompleksne fizikalne simulacije poput SPH metode. Daljnji koraci uključuju detaljnu implementaciju svakog od ovih aspekata kako bi se postigla realistična simulacija fluida.

4.3. Čestica

Klasa `Particle` sadrži sve potrebne atribute i metode za simulaciju ponašanja čestica fluida. Klasa se temelji na Unityjevom `MonoBehaviour` sustavu što omogućava lako povezivanje sa Unity komponentama i metodama životnog ciklusa.

Klasa sadrži mnogo klasnih varijabli, no ovo su najbitnije

- **float radius:** radijus čestice, ne želimo pre velike čestice jer će simulacija vode izgledati heterogeno, ali ne želimo ni premalene, jer će biti ili pre malo fluida, ili će simulacija biti spora zbog prevelikog simulacijskog napora
- **Vector2 position:** Pozicija čestice prikazana pomoću Unityjeve klase Vector2, ovime spremamo x i y poziciju čestice na sceni, Vector2 klasa dodatno omogućava lako računanje s drugim vektorima te znatno olakšava rad nad vektorima umjesto nad dvije varijable x i y.
- **Vector2 previousPosition:** Neke računice gledaju i prošlu poziciju neke čestice pa je bitno spremiti i prošlu poziciju radi toga
- **Vector2 velocityVector:** U ovu varijablu spremamo smjer i iznos brzine kretanja svake čestice
- **int gridX, gridY:** U ove varijable sprema se pozicija čestice unutar simulacijske mreže radi brzog traženja susjeda

4.4. Gustoća

Gustoća je ključni parametar u SPH simulaciji fluida, jer na temelju nje izračunavamo druge važne fizikalne veličine kao što su pritisak i viskoznost. Ovdje ćemo opisati metodu `ApplyDensity` koja se koristi za izračunavanje gustoće svake čestice u simulaciji.

Metoda `ApplyDensity` prolazi kroz sve čestice u simulaciji i za svaku česticu izračunava gustoću i susjednu gustoću na temelju udaljenosti od susjednih čestica. Susjedna gustoća jače drži na okupu one čestice koje su već blizu. Također, ova metoda identificira susjedne čestice i pohranjuje ih u listu susjeda čestice.

- **Prolazak kroz čestice:** Metoda započinje prolaskom kroz sve čestice u simulaciji pomoću `foreach` petlje. Svaka čestica (`particle`) prolazi kroz proces izračuna gustoće jednom po koraku simulacije.
- **Prolazak kroz ćelije mreže:** Za svaku česticu, metoda prolazi kroz sve ćelije u mreži koje okružuju trenutnu ćeliju čestice. Ovo se postiže dvostrukom `for` petljom koja prolazi kroz ćelije unutar radijusa od 1 ćelije od trenutne pozicije čestice (`gridX` i `gridY`).
- **Provjera granica mreže:** Prvi korak u tim petljama je provjera jesu li indeksi ćelija unutar granica mreže. Ovo osigurava da metoda ne pokušava pristupiti ćelijama koje su izvan granica definirane mreže (`xGridSize` i `yGridSize`).
- **Prolazak kroz čestice unutar ćelije:** Za svaku ćeliju unutar granica simulacije metoda će uzeti sve čestice u njima te napraviti provjeru sa trenutnom. Ako je čestica (`particle2`) ista kao i trenutna čestica (`particle`), preskače se daljnja obrada za tu česticu.
- **Izračun udaljenosti:** Metoda izračunava udaljenost između trenutne čestice i svake čestice u susjednim ćelijama pomoću metode `Vector2.Distance`.
- **Provjera radijusa:** Ako je udaljenost između čestica manja od predodređenog radijusa (`Config.R`), tada se čestica smatra susjedom.
- **Izračun normalizirane udaljenosti:** Normalizirana udaljenost se izračunava oduzimanjem omjera stvarne udaljenosti i radijusa od 1. Ovaj omjer se koristi za izračunavanje gustoće.
- **Ažuriranje gustoće:** Gustoća (`rho`) se povećava za kvadrat normalizirane udaljenosti, dok se susjedna gustoća (`rhoNear`) povećava za kub normalizirane udaljenosti. Ove vrijednosti se koriste za izračun pritiska u kasnijim koracima simulacije.
- **Dodavanje susjeda:** Čestica koja je unutar radijusa se dodaje u listu susjeda trenutne čestice (`neighbours`). Ova lista će se koristiti za daljnje proračune sila između susjednih čestica.

Metoda `ApplyDensity` je ključna za pravilno funkcioniranje SPH simulacije. Točan

izračun gustoće i susjedne gustoće omogućava pravilno modeliranje međučestičnih sila, što rezultira realističnim simulacijama tekućina. Identifikacija susjednih čestica i izračun njihovih doprinosa gustoći osigurava da svaka čestica ispravno reagira na svoje okruženje, omogućujući tekućini da teče i ponaša se kao stvarna tekućina.

4.5. Pritisak

Pritisak je ključan faktor u SPH simulacijama fluida, jer omogućava česticama da se međusobno odbijaju i održavaju strukturu tekućine. Ovdje ćemo opisati metodu `ApplyPressure` koje se koriste za izračunavanje i primjenu pritiska među česticama.

- **Prolazak kroz čestice:** Metoda započinje prolaskom kroz sve čestice u simulaciji pomoću `foreach` petlje. Za svaku česticu, izračunava se ukupna sila pritiska koja djeluje na nju.
- **Prolazak kroz susjede:** Unutar svake čestice, metoda prolazi kroz sve susjedne čestice pohranjene u listi `neighbours`. Ovo omogućava izračun međusobnih sila između susjednih čestica.
- **Izračun vektora razlike:** Vektor razlike između pozicija trenutne čestice i susjedne čestice (`vectorDiff`) izračunava se kako bi se dobio smjer sile pritiska.
- **Izračun udaljenosti:** Udaljenost između trenutne čestice i susjedne čestice izračunava se pomoću metode `Vector2.Distance`. Ova udaljenost koristi se za normalizaciju sile pritiska.
- **Normalizirana udaljenost:** Normalizirana udaljenost izračunava se oduzimanjem omjera stvarne udaljenosti i radijusa (`Config.R`) od 1. Ova normalizirana udaljenost koristi se za skaliranje sile pritiska.
- **Izračun ukupnog pritiska:** Ukupni pritisak (`pressureTotal`) izračunava se kao zbroj pritiska trenutne čestice i susjedne čestice, pomnožen sa kvadratom normalizirane udaljenosti, te zbroj susjednog pritiska trenutne i susjedne čestice, pomnožen sa kubom normalizirane udaljenosti.
- **Vektor sile pritiska:** Vektor sile pritiska (`pressureVector`) izračunava se mno-

ženjem ukupnog pritiska i normaliziranog vektora razlike pozicija čestica. Ova sila dodaje se sili susjedne čestice (`particleNeighbour.force`) i oduzima od sile trenutne čestice (`particle.force`).

- **Ažuriranje sile:** Nakon prolaska kroz sve susjedne čestice, ukupna sila pritiska oduzima se od sile trenutne čestice, osiguravajući da sila pritiska djeluje u ispravnom smjeru.

4.6. Viskoznost

Viskoznost je važan faktor u SPH simulacijama fluida, jer doprinosi unutarnjem trenju tekućine, omogućujući česticama da se kreću zajedno kao zajednička masa.

- **Prolazak kroz čestice:** Metoda započinje prolaskom kroz sve čestice u simulaciji pomoću `foreach` petlje. Za svaku česticu, izračunavaju se sile viskoznosti između nje i njezinih susjeda.
- **Prolazak kroz susjede:** Unutar svake čestice, metoda prolazi kroz sve susjedne čestice pohranjene u listi `neighbours`. Ovo omogućava izračun međusobnih sila viskoznosti između susjednih čestica.
- **Izračun vektora razlike:** Vektor razlike između pozicija trenutne čestice i susjedne čestice (`vectorDiff`) izračunava se kako bi se dobio smjer sile viskoznosti.
- **Izračun udaljenosti:** Udaljenost između trenutne čestice i susjedne čestice izračunava se pomoću metode `Vector2.Distance`. Ova udaljenost koristi se za normalizaciju sile viskoznosti.
- **Normalizacija vektora:** Vektor razlike se normalizira kako bi se dobio jedinični vektor smjera (`normalizedVectorDiff`).
- **Izračun relativne udaljenosti:** Relativna udaljenost izračunava se kao omjer stvarne udaljenosti i radijusa (`Config.R`).
- **Izračun razlike brzine:** Razlika brzine između trenutne čestice i susjedne čestice izračunava se kao skalarni produkt razlike brzine (`particle.velocityVector -`

`particleNeighbour.velocityVector`) i normaliziranog vektora smjera (`normalizedVectorDif`
Ovo daje komponentu razlike brzine duž smjera vektora razlike pozicija.

- **Primjena viskoznosti:** Ako je razlika brzine pozitivna (čestica se kreće prema susjednoj čestici), izračunava se sila viskoznosti. Sila viskoznosti je proporcionalna normaliziranoj udaljenosti, razlici brzine i konstanti viskoznosti (`Config.SIGMA`). Ova sila primjenjuje se na brzine trenutne čestice i susjedne čestice, smanjujući razliku u njihovim brzinama i simulirajući učinak unutarnjeg trenja.

- **Ažuriranje brzina:** Brzina trenutne čestice smanjuje se za polovicu izračunate sile viskoznosti, dok se brzina susjedne čestice povećava za istu vrijednost. Ovaj simetrični pristup osigurava da se ukupna količina gibanja sačuva, a čestice se približavaju zajedničkoj brzini.

4.7. Rezultantna sila

Literatura

[1] Microsoft, “C sharp (c)”, 2020. [Mrežno]. Adresa: https://iconduck.com/icons/27153/c-sharp-c

[2] U. Technologies, “The official unity logo”, 2021. [Mrežno]. Adresa: https://en.wikipedia.org/wiki/File:Unity_2021.svg

[3] C. Coetzee, “Review: Calibration of the discrete element method”, 2017. [Mrežno]. Adresa: https://www.sciencedirect.com/science/article/pii/S0032591017300268

[4] S. Liu, B. Wang, i X. Ban, “A symmetric particle-based simulation scheme towards large scale diffuse fluids”, 2018. [Mrežno]. Adresa: https://www.researchgate.net/figure/Schematics-of-two-methods-for-discretizing-the-flow-field-Grid-based-Euler-method_fig2_324097218

[5] D. Koschier, J. Bender, B. Solenthaler, i M. Teschner, “A survey on sph methods in computer graphics”, 2022. [Mrežno]. Adresa: https://animation.rwth-aachen.de/media/papers/77/2022-CGF-STAR_SPH.pdf

[6] X. Xia i Q. Liang, “Uniform grid searching method”, 2016. [Mrežno]. Adresa: https://www.researchgate.net/figure/Uniform-grid-searching-method_fig1_283526896

[7] H. Hristov, “K-d tree”, 2023. [Mrežno]. Adresa: https://www.baeldung.com/wp-content/uploads/sites/4/2023/03/kdtree.png

Sažetak

Vizualizacija dinamike fluida metodom hidrodinamike zaglađujućih čestica

Hrvoje Hemen

sažetak na hrvatskom

Ključne riječi: ključne riječi na hrvatskom

Abstract

Dynamic fluid visualization using smoothed particle hydrodynamics method

Hrvoje Hemen

abstract in English

Keywords: keywords in English