Alma Mater Studiorum - Università degli Studi di Bologna
Campus di Cesena

Scuola di Scienze
Corso di Studi in Ingegneria e Scienze Informatiche

Simulazione di fluidi in Computer Graphics

Relazione finale in

COMPUTER GRAPHICS

Relatore:

Dott.ssa Damiana Lazzaro

Presentata da:

Giacomo Ravaioli

Sessione III

Anno accademico 2014/2015

Parole chiave

Fluidodinamica computazionale
Approccio lagrangiano
Approccio Euleriano
Metodo Navier-Stokes
Metodo Lattice Boltzmann
Metodo Smoothed Particles Hydrodynamics

Indice

Introduzione

La simulazione di fluidi al calcolatore è uno degli argomenti maggiormente trattati e studiati nell'ambito della *computer graphics*. La corretta simulazione di fluidi sia newtoniani che non, permette di osservare gli effetti che alcuni eventi causerebbero sulle zone interessate. Si può pensare, ad esempio, al crollo di una diga o ad uno tsunami. La rappresentazione realistica di questi eventi al calcolatore, può aiutare gli ingegneri nel miglioramento delle tecniche di protezione e prevenzione. La simulazione di fluidi è presente in modo intensivo anche all'interno del settore videoludico e cinematografico, permettendo la realizzazione sia di ambienti realistici che di scene difficilmente riproducibili su un set, come ad esempio per il film "Poseidon" [MPC16]. Essendo largamente utilizzata in molti settori, compreso quello industriale, questa branca della computer graphics è fonte di continuo studio, soprattutto per la definizione di algoritmi sempre più efficienti per simulare in modo realistico e preciso i fluidi. L'implementazione di algoritmi efficienti, infatti, risulta molto difficile a causa della naturale complessità delle leggi matematiche e fisiche che ne governano il comportamento. Attualmente, esistono numerosi testi monotematici, gran parte dei quali non di recente pubblicazione, che trattano i singoli metodi senza però fornire tecniche implementative e/o valutazioni sui costi computazionali e di rendering.

Questa tesi si focalizza sullo studio dei modelli fisico-matematici attualmente in uso per la simulazione di fluidi al calcolatore con l'obiettivo di fornire nozioni di base e avanzate sull'utilizzo di tali metodi. La trattazione degli aspetti, sia teorici che implementativi, ha lo scopo di facilitare la comprensione dei principi su cui si fonda la simulazione di fluidi e rappresenta una base per la creazione di un proprio simulatore. E' possibile studiare le caratteristiche di un fluido in movimento mediante due approcci diversi, l'approccio lagrangiano e l'approccio euleriano. Mentre l'approccio lagrangiano ha lo scopo di conoscere la posizione occupata nel tempo o il valore, nel tempo, di una qualsiasi proprietà di ciascuna particella che compone il fluido, l'approccio euleriano, fissato uno o più punti del volume di spazio occupato da quest'ultimo, vuole studiare quello che accade, nel tempo, in quei punti. In particolare, questa tesi approfondisce, sia dal punto di vista numerico che del continuo, lo studio delle equazioni di Navier-Stokes, approcciandosi al problema in maniera euleriana. La soluzione numerica del sistema di

equazioni differenziali alle derivate parziali derivante dalle equazioni sopracitate, approssima la velocità del fluido, a partire dalla quale è possibile risalire a tutte le grandezze che lo caratterizzano. Attenzione, senza entrare molto nel dettaglio, viene riservata anche ad un modello facente parte dell'approccio semi-lagrangiano, il Lattice Boltzmann, considerato una via di mezzo tra i metodi puramente euleriani e quelli lagrangiani, che si basa sulla soluzione dell'equazione di Boltzmann mediante modelli di collisione di particelle. Infine, analogamente al metodo di Lattice Boltzmann, viene trattato il metodo Smoothed Particles Hydrodynamics, tipicamente lagrangiano, secondo il quale solo le proprietà delle particelle comprese dentro il raggio di una funzione kernel, centrata nella particella di interesse, influenzano il valore della particella stessa. Un resoconto pratico della teoria trattata viene dato mediante delle simulazioni realizzate tramite il software Blender 2.76b.

La tesi è così organizzata:

- Il capitolo 1 introduce il problema della simulazione dei fluidi al calcolatore, gli approcci possibili per risolverlo e l'introduzione al metodo di Navier-Stokes.
- Il capitolo 2 tratta esclusivamente del suddetto metodo, cuore della tesi. La natura duale di questa branca della computer graphics si ritrova anche in questo capitolo, che affronta il metodo sia dal punto di vista algebrico che dal punto di vista geometrico, fornendo soluzioni a problemi di tale natura così come tecniche implementative.
- Il capitolo 3 entra nello studio di un metodo il cui approccio alla natura del fluido è misto, permettendo di risolvere i problemi di dissipazione derivanti dal capitolo precedente. Tale capitolo affronta, inoltre, la simulazione di un fluido quale l'acqua.
- Il capitolo 4 si occupa di tecniche più complesse che permettono di aggiungere, alla simulazione, effetti o proprietà del fluido stesso. Il capitolo si conclude con la trattazione delle acque basse e degli oceani.
- Il capitolo 5 riguarda il progetto legato alla rappresentazione degli oceani proposta nel capitolo precedente. E' possibile, tuttavia, vedere numerose analogie con tutti gli argomenti trattati nei capitoli precedenti.
- Il capitolo 6 espone i metodi Lattice Boltzmann e Smoothed Particles Hydrodynamics per la simulazione dei fluidi e i relativi progetti.

Capitolo 1

1 I fluidi al calcolatore

Con il termine fluidi si definiscono tutti i materiali che si deformano illimitatamente se sottoposti a uno sforzo di taglio. Tale stato della materia comprende i liquidi, gli aeriformi, il plasma e, in alcuni casi, i solidi plastici.

Gli sforzi generati dalla deformazione dei fluidi dipendono principalmente dalla velocità con la quale si deformano e la relazione che intercorre tra queste due quantità ci permette di distinguere fra fluidi *newtoniani* e fluidi *non-newtoniani*. Per i primi, gli sforzi generati da una deformazione sono direttamente proporzionali alla velocità di deformazione stessa, mentre per i secondi intercorre una proporzionalità inversa.

Il comportamento dei fluidi è descritto da una serie di *equazioni differenziali a derivate parziali*, basate sulle leggi di conservazione della massa, del bilancio della quantità di moto e di conservazione dell'energia.

Nel paragrafo seguente viene trattata la legge che governa i fluidi newtoniani, non è infatti nell'interesse di questa tesi lo studio della simulazione al calcolatore di fluidi non-newtoniani. Nei paragrafi successivi del capitolo vengono poi esposti gli approcci che si possono adottare nello studio del moto dei fluidi ed il loro collegamento.

1.1 L'equazione fondamentale dei fluidi newtoniani

Il comportamento di gran parte dei fluidi di interesse alla computer graphics è descritto, dal punto di vista macroscopico, dalle *equazioni di Navier-Stokes*, ovvero un sistema di equazioni differenziali alle derivate parziali.

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} + \frac{1}{\rho} \nabla p = \vec{g} + \nu \nabla \cdot \nabla \vec{u} \tag{1.1}$$

$$\nabla \cdot \vec{u} = 0 \tag{1.2}$$

Nelle equazioni sopra riportate, $\vec{u} = (u, v, w)$ è la velocità del fluido, ρ la sua densità, p la pressione che esercita, $\vec{g}$ l'accelerazione dovuta alla forza di gravità e ν è la sua viscosità cinematica. Le equazioni di Navier-Stokes debbono il loro nome a Claude-Louis Navier e George Grabriel Stokes i quali le formalizzarono nel 1821 e la loro risoluzione analitica generale rappresenta tutt'ora uno dei problemi irrisolti della matematica moderna. La soluzione delle equazioni, seppur approssimata attraverso un metodo

numerico, fornisce il campo delle velocità del fluido, dal quale è possibile risalire a tutte le grandezze che lo caratterizzano.

Il modello matematico utilizzato per l'analisi della dinamica dei continui deformabili è il *modello del continuo* il quale si formula a partire da tre considerazioni: il fluido deve essere continuo, ovvero viene trascurata la natura discontinua della materia in modo da poter far tendere a zero un volume di fluido senza che questo rimanga privo di materia; il fluido deve essere chimicamente omogeneo e non reagente; il fluido deve essere privo di cariche. Tale modello risulta fondamentale per la formulazione delle equazioni (1.1) e (1.2).

1.1.1 L'equazione di incomprimibilità

L'equazione (1.2) è definita *equazione di incomprimibilità* ed è una condizione importante che permette di considerare il fluido da rappresentare incomprimibile, ovvero il cui volume non cambia nel tempo. Nella realtà i fluidi tendono a subire piccole perturbazioni di volume, tuttavia è impossibile generare una variazione importante dello stesso attraverso, ad esempio, l'utilizzo di presse idrauliche etc. Potrebbe essere di interesse la simulazione della piccola perturbazione sopra citata, però risulta altamente difficile e di dispendiosa realizzazione in relazione all'effetto prodotto poco visibile nella realtà. La condizione di incomprimibilità risulta quindi essenziale per la semplificazione della simulazione di un fluido attraverso l'utilizzo delle equazioni di Navier-Stokes.

Richiedere che il volume di un fluido non cambi nel tempo equivale a chiedere che la relativa velocità di variazione sia uguale a zero. Sia Ω il volume di un fluido con superficie di contorno $\partial\Omega$, è possibile misurare la velocità di variazione di Ω rispetto a $\partial\Omega$ integrando la componente normale della velocità del fluido su $\partial\Omega$.

$$ {}^{d}/_{dt}\, volume(\Omega) = \iint_{\partial\Omega} \vec{u} \cdot \hat{n} \tag{1.3} $$

La condizione di incomprimibilità si ottiene quindi ponendo la (1.3) uguale a zero.

$$ \iint_{\partial\Omega} \vec{u} \cdot \hat{n} = 0\ . \tag{1.4} $$

Utilizzando il teorema della divergenza ovvero una generalizzazione del teorema del calcolo fondamentale sulla (1.4) otteniamo

$$ \iiint_{\Omega} \nabla \cdot \vec{u} = 0\ . \tag{1.5} $$

L'equazione (1.5) deve risultare vera qualunque sia la regione di fluido Ω preso in considerazione, conseguentemente, l'unica funzione continua integrabile a zero scelto Ω è essa stessa zero. Da ciò l'equazione iniziale (1.2)

$$\nabla \cdot \vec{u} = 0. \tag{1.6}$$

Un campo vettoriale che soddisfa le condizioni di incomprimibilità si definisce *senza divergenza* o più comunemente *a divergenza nulla*.

1.1.2 L'equazione del moto

L'equazione (1.1) è *l'equazione del moto* la cui formulazione incomincia a partire dal secondo principio della dinamica, la legge di Newton. In natura ogni materiale è composto da un numero finito di particelle molto elevato. Si può, quindi, considerare il fluido composto da un insieme di quest'ultime aventi una massa *m*, un volume *V*, una velocità $\vec{u}$ e un'accelerazione descritta dal secondo principio della dinamica, per cui

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

Si considerino le forze che agiscono sul fluido e sulle particelle: la forza gravitazionale $\vec{F} = m\vec{g}$ esercitata da ogni particella, la pressione esercitata dal fluido e la sua viscosità. Tralasciando la forza gravitazionale, di banale intuizione, di maggior interesse risulta la pressione che svolge un ruolo di primaria importanza nelle dinamiche di un fluido. Regioni con alta pressione tendono a spingere regioni aventi bassa pressione, conseguentemente la componente dell'accelerazione relativa a tale forza influenza il comportamento del fluido solo nel caso in cui sia presente una variazione di pressione all'interno del fluido stesso. Per calcolare tale variazione di pressione è necessario l'utilizzo del gradiente negativo di quest'ultima, $-\nabla p$, ovvero il vettore indicante la direzione di maggior variazione di una data quantità in un dato punto del fluido. L'integrazione del gradiente su tutto il volume, approssimabile al prodotto tra questi, permette di ottenere l'intensità della forza esercitata dalla pressione.

$$\vec{F}_{pressione} = -V\nabla p.$$

La seconda forza che agisce sul fluido è generata dalla sua viscosità. Un fluido viscoso tende a resistere alle deformazioni che esso subisce; la forza dovuta alla viscosità tende infatti ad agire sulle particelle minimizzando la differenza di velocità tra queste. L'operatore differenziale che ci permette di misurare la differenza, in valore, di una certa quantità in un dato punto rispetto ai punti vicini è *l'operatore di Laplace* $\nabla \cdot \nabla$. Definendo μ il coefficiente di viscosità dinamica e integrando rispetto al volume, è possibile calcolare l'intensità della forza ad essa relativa. L'approssimazione dell'integrazione come il prodotto tra il volume e la data quantità, applicato anche precedentemente, permette di definire la forza sopracitata come

$$\vec{F}_{viscosità} = V\mu\nabla \cdot \nabla\vec{u}.$$

Si può scrivere, quindi, l'equazione del moto come

$$m\vec{a} = m\vec{g} - V\nabla p + V\mu\nabla \cdot \nabla\vec{u}. \quad (1.7)$$

Tale formulazione non rispetta, tuttavia, il modello matematico del continuo utilizzato per l'analisi della dinamica dei fluidi in quanto, relativamente a tale modello, la massa e il volume della particella tenderebbero a zero, facendo perdere di significato l'equazione. Si utilizzi quindi la definizione di pressione come $\rho = m/V$, ottenendo dopo banali sostituzioni nella (1.7)

$$\rho\vec{a} = \rho\vec{g} - \nabla p + \mu\nabla \cdot \nabla\vec{u}. \quad (1.8)$$

Si definisca ora la viscosità cinematica come $\nu = \mu/\rho$. Dividendo per il volume e modificando l'equazione (1.8) si ottiene

$$\vec{a} + \frac{1}{\rho}\nabla p = \vec{g} + \frac{\mu}{\rho}\nabla \cdot \nabla\vec{u}. \quad (1.9)$$

L'equazione così ottenuta differisce da quella iniziale per l'utilizzo delle derivate materiali, punto cruciale nella comprensione dell'equazione di Navier-Stokes. Queste rappresentano infatti l'unione tra due punti di vista per tracciare il moto di un fluido, *Euleriano* e *Lagrangiano*. Data l'importanza di quest'ultimi per la comprensione dell'equazione (1.1), la formulazione della stessa verrà completata nei paragrafi seguenti dopo successiva introduzione degli argomenti poc'anzi accennati.

1.2 Punto di vista lagrangiano – euleriano

Grande importanza per la comprensione e la formulazione delle equazioni di Navier-Stokes (1.1, 1.2) è data al punto di vista o sistema di riferimento tramite il quale si vuole analizzare il problema della simulazione dei fluidi. Questi sono: il punto di vista lagrangiano e il punto di vista euleriano.

Il punto di vista lagrangiano considera il fluido in quanto *sistema particellare*, ovvero un insieme di punti materiali su cui agiscono alcune leggi fisiche. Ogni punto o *particella* è identificata da un'etichetta ξ e da una posizione $\vec{x}$. Sia g una funzione vettoriale che fornisce la posizione della particella $\vec{x}$ all'istante di tempo t

$$\vec{x} = g(\xi, t), \quad (1.10)$$

allora è possibile definire la *velocità lagrangiana* della particella ξ come la derivata parziale rispetto a t della posizione $\vec{x}$ mantenendo costante ξ, ovvero

$$\vec{u} = \frac{\partial g(\xi,t)}{\partial t}, \quad (1.11)$$

e analogamente *l'accelerazione lagrangiana* come

$$\vec{a} = \frac{\partial^2 g(\xi,t)}{\partial t^2}\,. \tag{1.12}$$

Le proprietà del fluido, quindi, saranno funzioni sia del tempo sia del fluido stesso.

Utilizzando un approccio differente, il punto di vista euleriano considera punti fissi nello spazio e osserva come le proprietà del fluido (densità, velocità, temperatura etc.) cambiano nel tempo in questi punti. Tali proprietà sono definite come funzioni dello spazio e del tempo, l'osservatore rimane legato ad un riferimento fisso e "immortala" il campo di tali proprietà a ciascun istante temporale, senza possedere informazioni sulle singole particelle.

Fissato un punto $\vec{x}$ dello spazio, si definisca v una funzione che fornisce il campo di velocità $\vec{u} = v(\vec{x}, t)$, tramite il quale è possibile ottenere la velocità della particella che nell'istante *t* si trova in posizione $\vec{x}$.

Si può ora definire la *velocità dal punto di vista euleriano* come

$$\dot{x} = \frac{d\vec{x}}{dt} = \frac{\partial g(\xi,t)}{\partial t} = v(\vec{x}\,, t), \tag{1.13}$$

e conseguentemente *l'accelerazione euleriana* come

$$\ddot{x} = \frac{dv}{dt} = \frac{\partial^2 g(\xi,t)}{\partial t^2} = a(\vec{x}\,, t). \tag{1.14}$$

L'utilizzo di uno dei due punti di vista è arbitrario, tuttavia nell'ambito della fluidodinamica e della simulazione dei fluidi al calcolatori si predilige l'utilizzo del punto di vista euleriano il quale permette di lavorare analiticamente e di approssimare numericamente, in modo più semplice, le derivate spaziali quali il gradiente della pressione e la viscosità.

Nel seguito della tesi viene descritto in modo dettagliato l'utilizzo, a partire dal punto di vista euleriano, di un metodo che fa uso, anche se solo in modo parziale, del secondo punto di vista e di metodi che utilizzano esclusivamente l'approccio lagrangiano. Il collegamento tra i due punti di vista appena introdotti è dato da uno speciale strumento matematico: la *derivata materiale.*

1.3 Derivata materiale

La *derivata materiale* o *derivata lagrangiana* è una derivata totale che indica, in generale, una variazione nel tempo di una grandezza, riferita ad una specifica particella.

Si consideri il punto di vista lagrangiano, si hanno cioè un insieme di particelle aventi una posizione $\vec{x}$ e una velocità $\vec{u}$. Si indichi inoltre una proprietà della particella (velocità, densità etc.) con q. Ogni particella possiede un valore per questa proprietà che viene

calcolato dalla funzione $q(\vec{x}, t)$ sulla base della posizione della particella stessa ad un dato istante di tempo *t*. La variabile che rappresenta la proprietà considerata, essendo in funzione dello spazio e non della particella, è euleriana e per trovare quanto velocemente cambia per la particella in posizione $\vec{x}(t)$ è necessario svolgere la derivata totale.

$$\frac{d}{dt} q(\vec{x}(t), t) = \frac{\partial q}{\partial t} + \nabla q \cdot \frac{d\vec{x}}{dt} = \frac{\partial q}{\partial t} + \nabla q \cdot \vec{u} \equiv \frac{Dq}{Dt} \tag{1.15}$$

$$\frac{Dq}{Dt} = \frac{\partial q}{\partial t} + u\frac{\partial q}{\partial x} + v\frac{\partial q}{\partial y} + w\frac{\partial q}{\partial z} \tag{1.16}$$

Nell'equazione (1.15) il primo termine $\frac{\partial q}{\partial t}$, una misura euleriana, indica quanto velocemente varia q in un dato punto dello spazio, mentre il secondo termine mira a correggere eventuali errori dovuti alle perturbazioni del fluido.

L'equazione (1.15) detta anche *equazione di avvezione*, ovvero un'equazione che utilizza derivate materiali, ci indica come una proprietà cambia mentre si muove all'interno del campo vettoriale $\vec{u}$. Se tale equazione si annulla, cioè risulta uguale a zero, allora è possibile affermare che la proprietà *q* non varia mentre il fluido si muove.

Questa definizione risulta molto importante per poter collegare, come affermato in precedenza, i due punti di vista ed ottenere una formulazione completa dell'equazione di Navier-Stokes.

Per poter fare ciò, consideriamo l'accelerazione della particella come $\vec{a} \equiv \frac{D\vec{u}}{Dt}$, ovvero come la variazione della velocità della singola particella. Si può notare come questa notazione sia l'applicazione della derivata materiale alla proprietà $\vec{u}$, cioè la velocità della singola particella.

Si può allora riscrivere la (1.9) utilizzando quest'ultima notazione ottenendo

$$\frac{D\vec{u}}{Dt} + \frac{1}{\rho}\nabla p = \vec{g} + \nu\nabla \cdot \nabla\vec{u}. \tag{1.17}$$

Sapendo dalla definizione di derivata materiale che

$$\frac{Dq}{Dt} \equiv \frac{\partial q}{\partial t} + \nabla q \cdot \vec{u},$$

definendo quest'ultima per la velocità $\vec{u}$ si ottiene la (1.1)

$$\frac{\partial\vec{u}}{\partial t} + \vec{u} \cdot \nabla\vec{u} + \frac{1}{\rho}\nabla p = \vec{g} + \nu\nabla \cdot \nabla\vec{u}$$

1.4 Condizioni aggiuntive alle equazioni di Navier-Stokes

Alle equazioni di Navier-Stokes è necessario aggiungere condizioni che modellino ulteriori aspetti della simulazione di fluidi al calcolatore; le *condizioni ai bordi*. Le

condizioni ai bordi sono di primaria importanza, in quanto regolano le interazione fra il fluido e i confini che lo delimitano. Le due principali condizioni imponibili riguardano l'interazione del fluido con una superficie solida o con una superficie libera.

La presenza di un solido a contatto con un fluido fa sì che la velocità con cui questo scorre da/verso il primo, cioè la componente normale della velocità del fluido, sia pari a zero, ovvero

$$\vec{u} \cdot \hat{n} = 0. \tag{1.18}$$

L'equazione (1.18), però, è relativa al caso particolare in cui il solido non si muova. In generale è necessario che la componente normale della velocità del fluido eguagli la componente normale della velocità del solido, quindi

$$\vec{u} \cdot \hat{n} = \vec{u}_{solido} \cdot \hat{n}, \tag{1.19}$$

definita come condizione *no-stick*. Questa condizione è ovviamente propria di fluidi non viscosi i quali si muovono senza che la loro velocità venga alterata. Nel caso si fosse in presenza di fluidi viscosi, tutte le componenti della velocità verrebbero modificate, trasformando la (1.18) in

$$\vec{u} = 0,$$

e la (1.19) in

$$\vec{u} = \vec{u}_{solido}, \tag{1.20}$$

definita come condizione *no-slip*.

Infine, vi è la possibilità che il fluido possa allontanarsi dal confine o bordo, rendendo necessario l'utilizzo della disequazione

$$\vec{u} \cdot \hat{n} \geq \vec{u}_{solido} \cdot \hat{n},$$

la quale impone che la componente normale della velocità del fluido sia maggiore di quella del solido.

La seconda condizione ai bordi, riguardante le superfici libere, richiede l'utilizzo di un modello matematico adeguato. Considerare l'aria come una regione con pressione atmosferica costante, permette di individuare una superficie libera solo quando p si mantiene costante, ad esempio uguale a zero, senza richiedere alcun controllo sulle velocità.

$$p = 0 \tag{1.21}$$

Si riprenderà lo studio delle condizioni ai bordi più avanti in questo documento, in quanto necessarie per capire in che rapporto un determinato punto dello spazio preso in considerazione sta con il fluido e i confini del sistema da simulare.

Capitolo 2

2 Metodo di Navier-Stokes

Il metodo di Navier-Stokes per la simulazione dei fluidi al calcolatore si basa sull'implementazione delle equazioni omonime descritte nel capitolo precedente. In questo capitolo viene trattata la suddetta implementazione, preceduta da una fase di preparazione all'utilizzo del metodo con la discretizzazione del spazio, del tempo e delle equazioni (1.1) e (1.2). Questo capitolo possiede anche una natura geometrica intrinseca, in quanto in questa parte di trattazione vengono risolti problemi di vario tipo, di cui i più importanti da risolvere introdotti dalle condizioni ai bordi. Visti gli argomenti discussi in questa parte del testo, questo capitolo può essere considerato il cuore di un simulatore di fluidi al calcolatore, conseguentemente gli sarà data particolare enfasi.

2.1 Impostazione del metodo

Per simulare numericamente al calcolatore un fluido, descritto dalle equazioni (1.1) e (1.2), è necessario separarle nelle loro componenti base, in quanto l'implementazione diretta, richiederebbe la risoluzione delle stesse nella loro totalità, aumentando notevolmente la difficoltà implementativa e i costi computazionali. Durante la suddivisione delle equazioni di Navier-Stokes è necessario porre particolare attenzione alle condizioni nelle quali le equazioni risultanti andrebbero risolte. Condizione necessaria per un corretto svolgimento del metodo è quella di garantire le condizioni di incomprimibilità (1.2), cioè assicurarsi che il campo di velocità rimanga a divergenza nulla durante ciascuna iterazione del metodo.

La discretizzazione della (1.1) e della (1.2) avviene separandole nel seguente insieme di equazioni:

$$\frac{Dq}{Dt} = 0, \tag{2.1}$$

$$\frac{\partial \vec{u}}{\partial t} = \vec{g}, \tag{2.2}$$

$$\frac{\partial \vec{u}}{\partial t} + \frac{1}{\rho}\nabla p = 0 \tag{2.3a}$$

$$\nabla \cdot \vec{u} = 0. \tag{2.3b}$$

La (2.1) rappresenta l'equazione di avvezione di una data quantità *q* rispetto al campo di velocità $\vec{u}$ per un dato intervallo di tempo Δt. L'equazione (2.2) rappresenta le forze del corpo. Questa è una equazione differenziale ordinaria (ODE) e in quanto tale è possibile risolverla con il metodo di Eulero. Infine, le (2.3a) e (2.3b) rappresentano rispettivamente l'equazione della pressione e la condizione di incomprimibilità. Queste possono essere risolte attraverso un algoritmo che calcola ed applica la giusta pressione al campo di velocità, in modo da garantire che la convergenza di quest'ultimo sia nulla ed assicurare le condizioni ai bordi. Tale garanzia, necessaria per il calcolo dell'avvezione, permette di simulare correttamente il fluido e di evitare fenomeni problematici quali l'aggiunta o la perdita dello stesso e del suo moto.

Di seguito si mostra l'algoritmo che descrive i passaggi di una simulazione di fluidi non viscosi al calcolatore:

- L'algoritmo inizia con un campo di velocità iniziale $\vec{u}^0$ a divergenza nulla
- Per ogni iterazione $n = 0,1,2, \dots$
    - Si determina un intervallo di tempo Δt per passare da t_n a t_{n+1}
    - Si compie l'avvezione, risolvendo la (2.1), ottenendo un nuovo campo di velocità $\vec{u}^A$
    - Si calcolano le forze del corpo, risolvendo la (2.2), ottenendo il nuovo campo di velocità $\vec{u}^B = \vec{u}^A + \Delta t\vec{g}$
    - Si calcola e applica la giusta pressione al campo vettoriale per mantenerlo a divergenza nulla, ottenendo $\vec{u}^{n+1}$

La discretizzazione non riguarda solo le equazioni da risolvere, ma anche il tempo e lo spazio. La scelta dell'intervallo di tempo per il calcolo dell'avvezione e della pressione è molto importante e, pur essendo del tutto arbitraria, è necessario seguire diverse accortezze. Un intervallo di tempo consono dovrebbe rimanere inferiore alla durata di un *frame* e maggiore rispetto al minimo richiesto dalle varie parti risolutive delle equazioni (2.1), (2.2), (2.3a) e (2.3b). La scelta dell'intervallo di tempo adatto rimane comunque legata alla qualità della simulazione ricercata, ai requisiti di performance richiesti e al tempo medio di esecuzione degli algoritmi. Di conseguenza, nei vari paragrafi sarà trattata in modo opportuno la suddetta scelta.

Per la discretizzazione dello spazio, il metodo più utilizzato, per la risoluzione di problemi di flussi incomprimibili, è il *metodo Marker-And-Cell* (MAC), introdotto da F. Harlow e J. Welch [HW65]. L'idea alla base del metodo MAC è quella di utilizzare una *griglia MAC* sfalsata in cui le variabili vengono memorizzate in posizioni differenti. Data una cella della griglia (i, j), la pressione $p_{i,j}$ è memorizzata al suo interno mentre la velocità è separata nelle sue componenti normali rispetto alle facce della cella e memorizzata al centro di quest'ultime; ad esempio $v_{i,j+1/2}$ indica la velocità verticale tra le celle (i, j) e $(i, j+1)$.

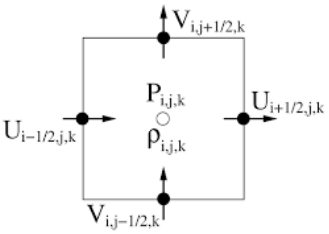


Figura 1- Cella di una griglia MAC 2D

Per calcolare il gradiente della pressione e la convergenza della velocità, a partire dai valori presenti nella griglia, è possibile utilizzare il metodo delle *differenze centrali*. L'uso di questo metodo applicato a griglie non sfalsate, valutando funzioni irregolari, può causare il cosiddetto problema degli spazi nulli non banali. La valutazione in un punto *i*, infatti, ignora il valore q_i, producendo stime di gradienti nulle. Questo motiva l'utilizzo di griglie sfalsate per cui la stima di $\frac{\partial q}{\partial t}$ al punto della griglia *i* è pari a

$$\left(\frac{\partial q}{\partial t}\right)_i \approx \frac{q_{i+1/2} - q_{i-1/2}}{\Delta x}. \tag{2.5}$$

Altro pregio che porta all'utilizzo di questa tipologia di griglie è il fatto che, al pari delle griglie non sfalsate, offrono un'accuratezza di $O(\Delta x^2)$.

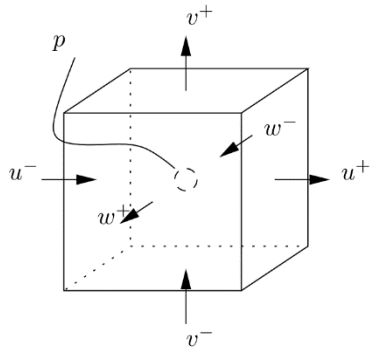


Figura 2 - Cella di una griglia MAC 3D

Le griglie MAC permettono, inoltre, di lavorare sia in 2D che in 3D, incrementando il numero di indici utilizzati e sostituendo a celle quadrate celle cubiche. Tuttavia, lavorare con griglie in 3D, introduce tre problemi: se la regione occupata da un fluido cambia significativamente nel corso della simulazione, l'uso di una griglia che racchiuda l'intera regione possibilmente occupabile dal fluido porterebbe ad un'inevitabile spreco di risorse; la scarsa vicinanza, nella memoria fisica, di punti vicini nella griglia comporta maggiori rischi di *page fault* e *cache miss*; se il fluido occupa una piccola regione rispetto al volume messo a disposizione, si avrebbe uno spreco elevato di risorse di calcolo.

L'uso di *griglie a blocchi sparsi* ovvero griglie di dimensione 2^{32} in ogni direzione, divise in blocchi di dimensione fissa memorizzati come vettori 3D contigui [BRI03, MUS11,

MUS13]. Ad ogni istante, gli unici blocchi allocati in una struttura dati associativa, che li mappa attraverso i rispettivi indici, sono quelli utilizzati e necessari ai fini della simulazione. L'utilizzo di una struttura dati sparsa avente una buona località dei dati permette, quindi, di risolvere i problemi sopra esposti.

Nei paragrafi seguenti paragrafi vengono trattate le tecniche implementative delle equazioni (2.1), (2.3a) e (2.3b).

2.2 Avvezione

Come introdotto nel paragrafo precedente, l'avvezione svolge un ruolo di importanza all'interno della simulazione dei fluidi e ci permette di ottenere l'approssimazione del nuovo valore, di una data quantità *q*, attraverso il campo di velocità $\vec{u}$ nell'intervallo di tempo Δt.

L'approccio utilizzato per la risoluzione della (2.1) è *semi-lagrangiano*, per questo l'avvezione viene definita con lo stesso termine [STA99]. Tale tecnica "alternativa" è resa necessaria dall'elevata instabilità del metodo di Eulero in avanti, utilizzabile se si volesse risolvere la (2.1) rappresentandola come equazione differenziale alle derivate parziale (EDP). Il metodo di Eulero in avanti per la risoluzione di EDP, infatti, risulta estremamente instabile per la risoluzione di derivate spaziali. A questo va aggiunto il problema degli spazi nulli presentato nel paragrafo precedente. Il termine semi-lagrangiano deriva dallo studio della velocità delle particelle e delle proprietà ad esse, tipico dei metodi lagrangiani, senza però la loro effettiva creazione.

Considerato il punto di vista lagrangiano, sia $\vec{x}_G$ la posizione nella quale si vuole calcolare il nuovo valore di *q* e sia q_G tale valore. Sia $\vec{x}_P$ la posizione di partenza della particella che, dopo l'intervallo di tempo Δt, si trova in $\vec{x}_G$ allora

$$q_G = q_P,$$

dove q_P è il suo valore in $\vec{x}_P$.

E' necessario, quindi, individuare la posizione nello spazio dalla quale è partita la suddetta particella, risolvendo l'ODE

$$\frac{d\vec{x}}{dt} = \vec{u}(\vec{x}).$$

Utilizzando uno step del *metodo Runge-Kutta* di secondo ordine (RK2) si ottiene

$$\vec{x}_{mid} = \vec{x}_G - \frac{1}{2}\Delta t\vec{u}(\vec{x}_G),$$

$$\vec{x}_P = \vec{x}_G - \frac{1}{2}\Delta t\vec{u}(\vec{x}_{mid}).$$

Risultati migliori possono essere ottenuti attraverso l'utilizzo dei metodi RK di ordine superiore al secondo come, ad esempio, RK3 o RK4.

Una volta ottenuta la posizione iniziale della particella si trova il valore q_P approssimato. Nel caso la particella fosse all'interno di una delle celle della griglia e non su una delle facce di queste, la sua posizione corrisponderebbe all'interpolazione dei valori vicini sulla griglia. L'operazione di interpolazione, tuttavia, produce una dissipazione dei valori interpolati, tendendo a renderli omogenei e producendo spesso un effetto di sfumatura non desiderato, accentuato dall'utilizzo di tecniche di interpolazione, quali quella lineare. Per evitare tale effetto, definito *diffusione numerica*, è necessario utilizzare tecniche di interpolazioni meno diffusive come, ad esempio, l'interpolazione di Catmull-Rom [FSJ01] o un interpolazione cubica, bi-cubica o tri-cubica.

Diversamente accade se la posizione iniziale della particella si trova in aree esterne al volume di simulazione del fluido per il quale non si hanno informazioni. Questo è il problema per cui vengono imposte le condizioni ai bordi e la cui risoluzione viene trattata nel paragrafo seguente.

Per concludere la trattazione sull'avvezione, si vuole studiare quali siano i valori adeguati di Δt, relativamente a tale parte del metodo, per ottenere una buona simulazione. Il metodo semi-lagrangiano poc'anzi esposto risulta incondizionatamente stabile per ogni valore di Δt, anche molto piccolo. Ciò è dato dal fatto che le operazioni di interpolazione producono risultati sempre compresi tra i valori su cui la si sta eseguendo. Tuttavia, una buona pratica per limitare Δt ed evitare risultati inaspettati è stata introdotta da Foster e Fekiw [FF01], i quali suggeriscono di porre

$$\Delta t \leq \frac{5\Delta x}{u_{max}}, \tag{2.6}$$

dove u_{max} è una stima della velocità massima del fluido.

Questa condizione permette di limitare Δt, dimodoché la traiettoria più lunga tracciata da una particella, nel suddetto intervallo, sia al massimo un numero costante di celle, ad esempio cinque. Per il valore di u_{max} è possibile utilizzare un semplice dato memorizzato in quel momento nella griglia, oppure una stima più robusta che considera anche le velocità indotte, ad esempio, dall'accelerazione gravitazionale g. In tal caso

$$u_{max} = \max(|u^n|) + \Delta t|g|,$$

sostituendo in questa relazione Δt con l'estremo superiore della (2.6) si ottiene

$$\mathrm{u}_{\max} = \max(|u^n|) + \frac{5\Delta \mathrm{x}}{\mathrm{u}_{\max}}|g|,$$

che risolta per $\mathrm{u}_{\max}$ restituisce

$$\mathrm{u}_{\max} = \max(|u^n|) + \sqrt{5\Delta \mathrm{x} g}. \tag{2.7}$$

Prima di continuare con le tecniche implementative delle equazioni (2.3a) e (2.3b), si vuole introdurre un metodo che permette di dare una risposta adeguata ai problemi sorti in questo paragrafo e per fare ciò, la trattazione assumerà una connotazione prevalentemente geometrica.

2.3 Level Set Method: risoluzione dei problemi geometrici delle condizioni ai bordi

Nel paragrafo precedente, si è visto come il metodo semi-lagrangiano, nel tracciare il punto di partenza di una particella, introduca problemi geometrici legati alle condizioni ai bordi. Tra questi si possono individuare i seguenti: capire se un punto è interno ad un solido, trovare il punto più vicino, sulla superficie di un data geometria, ad un altro e estrapolare valori da una regione ad un'altra.

Questi problemi si possono risolvere con l'utilizzo di una rappresentazione della geometria che si rapporti, in modo consono, alla discretizzazione delle equazioni su una griglia come, ad esempio, il *level set method* (LSM) o *metodo degli insieme dei livelli*. LSM è stato sviluppato nel 1980 dai matematici americani Stanley Osher e James Sethian [OF02] e utilizza il concetto di insieme di livello come strumento per l'analisi numerica di superfici.

2.3.1 Signed distance

Sia $\phi(x, y, z)$ una funzione scalare continua che definisce la geometria di una superficie implicita nel seguente modo:

- un punto $\vec{\mathrm{x}}$ è all'esterno della geometria se $\phi(\vec{\mathrm{x}}) > 0$,
- è interno alla geometria se $\phi(\vec{\mathrm{x}}) < 0$, e
- un punto $\vec{\mathrm{x}}$ è sulla superficie quando $\phi(\vec{\mathrm{x}}) = 0$.

Se $\phi(\vec{\mathrm{x}})$ è differenziabile su di un punto sulla superficie, allora $\nabla\phi(\vec{\mathrm{x}})$ rappresenta il vettore gradiente in quel punto e punta verso l'esterno della superficie. Per i calcoli successivi, è necessario che il vettore gradiente abbia esattamente lunghezza unitaria, cioè

$$\|\nabla\, \phi(\vec{\mathrm{x}})\| = 1, \tag{2.8}$$

ciò, tenendo conto del fatto che questo risulta parallelo alla normale, equivale a imporre che la derivata direzionale di ϕ, nella direzione normale $\hat{n}$, sia esattamente uno.

$$\frac{\partial \phi(\vec{x})}{\partial \hat{n}} = 1 \tag{2.9}$$

La (2.9), per valori molto piccoli di *d* e relativamente ad un punto $\vec{x}$, risulta

$$\frac{(\phi(\vec{x}+d\hat{n}) - \phi(\vec{x}))}{d} \approx 1 \text{ allora } \phi(\vec{x} + d\hat{n}) - 0 \approx d, \tag{2.10}$$

dove *d* è la distanza dalla superficie sulla quale si sta valutando ϕ.

La (2.10) rappresenta un risultato molto importante in quanto permette di affermare che la distanza di un punto dalla superficie è data dalla valutazione della funzione ϕ in quel punto.

Tale funzione prende il nome di *funzione delle distanze orientate* o *signed distance function* (SDF).

Dato un insieme S di punti, è possibile definire la *funzione distanza* per l'insieme S come

$$\text{distance}_S(\vec{x}) = \min_{\vec{p} \in S} \|\vec{x} - \vec{p}\|$$

e, nel caso quest'ultimo suddivida lo spazio in sottospazi ben definiti (interno e esterno), si può definire la sua SDF come

$$\phi_S(\vec{x}) = \begin{cases} \text{distance}_S(\vec{x}) & : \vec{x} \text{ è esterno} \\ -\text{distance}_S(\vec{x}) & : \vec{x} \text{ è interno} \end{cases}$$

L'uso delle SDF permette di affermare che, per un punto $\vec{x}$ esterno alla geometria della superficie, $\vec{p}$ rappresenta il punto più vicino ad esso sulla superficie stessa e anche che $-\nabla \phi(\vec{x})$ punta sempre verso quest'ultimo.

Sia $\hat{c}$ la direzione verso il punto più vicino ad $\vec{x}$ sulla superficie, cioè

$$\hat{c} = \frac{\vec{p} - \vec{x}}{\|\vec{p} - \vec{x}\|},$$

allora $\nabla \phi(\vec{x}) = -\hat{c}$.

Sapendo, infine, che il valore di ϕ corrisponde alla distanza tra il un punto $\vec{x}$ e quello ad esso più vicino sulla superficie, è possibile affermare che

$$\vec{p} = \vec{x} - \phi(\vec{x}) \nabla \phi(\vec{x}) \, . \tag{2.11}$$

Anche se lo studio di come, dato un punto, si possa trovare il valore di ϕ viene affrontato nei paragrafi seguenti, è necessario specificare che, includendo operazioni di interpolazione, nel caso in cui si verifichi un errore durante il calcolo, la (2.11) potrebbe restituire un punto non necessariamente sulla superficie. Sarebbe buona pratica, quindi, affiancare a quest'ultima delle procedure iterative per ottenere risultati più affidabili.

Le SDF possono essere definite anche a partire dalla (2.8), nel caso in cui siano imposte le appropriate condizioni ai bordi e siano specificate le informazioni tecniche riguardanti il flusso del fluido. Nel corso del paragrafo verrà introdotta una tecnica per calcolare le distanze orientate mediante la risoluzione di tale PDE.

2.3.2 Discretizzazione di SDF: Level Set Method

Calcolare analiticamente le SDF, a partire da informazioni sulla geometria degli oggetti può essere un approccio utile nel caso quest'ultimi siano molto semplici. Molto spesso, però, è necessario definire le SDF per oggetti complessi di cui si potrebbe non aver alcuna informazione geometrica a riguardo. Con il metodo LSM, invece di calcolare analiticamente la SDF, si memorizzano i valori delle distanze orientate all'interno della griglia MAC. La valutazione di $\phi(\vec{\mathrm{x}})$ risulterà dall'interpolazione dei valori presenti nelle celle circostanti attraverso l'utilizzo di metodi intuitivi, quali quello delle *differenze finite*. Ad esempio, la stima di $\frac{\partial \phi}{dx}$ sull'asse *x* tra due punti della griglia sarà data da

$$\left(\frac{\partial \phi}{dx}\right)_{i+\frac{1}{2},j,k} \approx \frac{\phi_{i+1,j,k}-\phi_{i,j,k}}{\Delta \mathrm{x}}.$$

Allo stesso modo, è possibile avere una stima di punti intermedi a valori della griglia relativamente agli altri due assi.
Da questa discretizzazione deriva il significato con cui viene più comunemente inteso un level set, ovvero, il campionamento su di una griglia di una SDF.

2.3.3 Calcolo di level set: approccio geometrico e approccio numerico

Il calcolo di un level set può avvenire in due modi: attraverso un *approccio geometrico*, cercando i punti più vicini alla geometria di una superficie e calcolando la distanza da essi oppure attraverso un *approccio alle PDE* e la risoluzione dell'equazione di Eikonal (2.8). Ogni approccio possiede i suoi pregi e i suoi difetti che verranno evidenziati nel corso del paragrafo, tuttavia, è possibile definire delle situazioni generali nelle quali è conveniente l'utilizzo di uno dei due rispetto all'altro.
Il calcolo di un level set attraverso l'approccio geometrico risulta, solitamente, più accurato e facile da capire, ma richiede la conoscenza delle informazioni sulla geometria della superficie come, ad esempio, le distanze orientate. In assenza di queste informazioni è necessario procedere utilizzando la seconda tecnica.
Come già specificato, se la geometria delle superfici è molto semplice oppure all'interno della griglia MAC sono presenti le informazioni riguardanti le distanze orientate,

l'approccio geometrico andrebbe preferito. In questa trattazione, tuttavia, per mostrare un metodo più completo e utilizzabile in ogni situazione si preferisce porre l'accento sull'approccio alle PDE. E' possibile trovare algoritmi efficienti per il calcolo della distanza orientata di un punto da un insieme di punti o da una *mesh* triangolare, nel caso si voglia utilizzare la prima tecnica, negli articoli di Y. Tsai [TSA02] e di M. Jones [JBS06].

L'utilizzo dell'approccio alle PDE prevede la risoluzione, come detto poc'anzi, dell'equazione (2.8). Per fare ciò, è necessario avere inizialmente una stima precisa della distanza dalla superficie alle celle nelle stesse per le quali si verifica un cambio di segno. Sia F la funzione iniziale, se nel punto della griglia (i,j,k) si ha che il segno di $F_{i,j,k}$ è diverso dal segno di $F_{i+1,j,k}$, allora si può interpolare F lungo il confine tra i due punti per i quali l'interpolazione lineare è pari a zero, cioè

$$\theta = \frac{F_{i,j,k}}{F_{i,j,k} - F_{i+1,j,k}}.$$

Si ponga $\phi_{i,j,k} = sign(F_{i,j,k})\theta\Delta x$ e si svolga tale operazione per tutti i vicini di (i,j,k) il cui segno di F varia. Una volta assegnato il valore di $\phi_{i,j,k}$ a tutti i punti vicini alla superficie, è necessario propagare la distanza a tutti gli altri della griglia. Tsai ha introdotto due metodi per poter svolgere la propagazione: il *fast marching method* (FMM) e il *fast sweeping method* (FSM). La differenza dei metodi risiede nel modo con il quale viene propagata l'informazione sulla distanza orientata: nel FMM l'informazione si propaga da un punto a tutti i suoi vicini; nel FSM l'informazione si propaga dai punti più vicini a quelli più lontani cioè, ad ogni punto, la propagazione arriva da una direzione specifica.

Tali metodi, nati per l'utilizzo nell'approccio geometrico, possono essere utilizzati anche per quello finora adottato, sostituendo il calcolo geometrico dei valori della distanza orientata con la risoluzione dell'equazione (2.8). Si riscriva questa come

$$\left(\frac{\partial\phi}{\partial x}\right)^2 + \left(\frac{\partial\phi}{\partial y}\right)^2 + \left(\frac{\partial\phi}{\partial z}\right)^2 = 1\,, \qquad (2.12)$$

per stimare il valore della distanza orientata basta sostituire nella (2.12) le derivate parziali con le differenze finite. Si consideri la stima di $\phi_{i,j,k}$ supponendo che la propagazione arrivi dai punti $\phi_{i-1,j,k}$, $\phi_{i,j+1,k}$ e $\phi_{i,j,k-1}$, allora si ottiene

$$\left(\frac{\phi_{i,j,k} - \phi_{i-1,j,k}}{\Delta x}\right)^2 + \left(\frac{\phi_{i,j,k} - \phi_{i,j+1,k}}{\Delta y}\right)^2 + \left(\frac{\phi_{i,j,k} - \phi_{i,j,k-1}}{\Delta z}\right)^2 = 1.$$

I level set e il metodo LSM costituiscono uno strumento essenziale per la risoluzione dei problemi relativi alla geometria delle superfici.

Grazie all'utilizzo dei level set, infatti, è possibile: determinare le zone interne e esterne ad una superficie, svolgendo una banale interpolazione di ϕ e valutandone i segni; ottenere informazioni sulla geometria della superficie o trovare, dato un punto $\vec{x}$, il punto $\vec{p}$ più vicino ad esso su di una superficie con la (2.11).

I level set permettono anche l'utilizzo di una serie di operazioni di notevole importanza, come ad esempio, l'uso del *ray-tracing* in fase di rendering, direttamente su di essi. Tale operazione dipende fortemente, però, dalla tecnica interpolante utilizzata, per questo si consiglia l'utilizzo di tecniche basate su B-Spline quadratiche. Per la simulazione realistica di fluidi, quali l'acqua, una proprietà importante dei level set è quella di poter essere sottoposti ad avvezione, analogamente ai campi di velocità. Se si vuole muovere una superficie lungo un campo di velocità $\vec{u}$, è necessario che i punti per i quali $\phi(\vec{x}) = 0$ si muovano seguendo

$$\frac{d\vec{x}}{dt} = \vec{u}.$$

In generale, ricordandosi la definizione di avvezione, si vuole che ogni punto del dominio si muova seguendo il campo $\vec{u}$, mantenendo il proprio valore ϕ. Ciò equivale a chiedere che la variazione di ϕ per ogni punto sia nulla, cioè

$$\frac{D\phi}{Dt} = 0.$$

Dopo un'operazione di avvezione sarà necessario, però, ricalcolare le distanze orientate per tutti i punti, effettuando tale aggiornamento ad ogni frame nel corso di una simulazione. Va aggiunto, inoltre, che l'avvezione di un level set risente pesantemente della diffusione numerica, perciò è necessario che sia applicata almeno un'interpolazione cubica.

Sempre per la rappresentazione di fluidi, ha grande importanza la possibilità di estrapolare dati da una regione della griglia ed estenderli ad altre. Nel paragrafo precedente, infatti, l'avvezione aveva introdotto il problema del calcolo del valore di una data quantità, una volta trovato il punto di partenza della particella in posizione $\vec{x}$, nel caso in cui la particella risultasse fuori dai confini del dominio nel quale si sta effettuando la simulazione L'estrapolazione di dati da una regione, permessa dall'uso dei level set, risolve il suddetto problema. Per fare ciò, si utilizza *la ricerca in ampiezza* o *Breadth-*

first search (BFS) ma, mentre i valori vicini alla superficie risultano ragionevoli, i più lontani tendono a generare errori nella simulazione.
Conseguentemente, se la necessità di estrapolare i dati riguarda valori vicini alla superficie, l'algoritmo BFS risulta il migliore, altrimenti, nel caso la SDF sia stata calcolata geometricamente, è possibile impostare il valore in un punto della griglia uguale a quello del punto più vicino nella geometria della superficie in input.

Infine, l'utilizzo di level set permette di effettuare operazioni minori ma di non meno importanza come l'applicazione di filtri di *sharpening* e *smoothing*, l'esecuzione di operazioni di *geometria solida booleana* o *costruttiva* e la ricostruzione di *mesh* di una superficie.
Concludendo il paragrafo è necessario, però, sottolineare che i level set possono essere usati solo per rappresentare superfici che non si intersecano, superfici che presentano aree interne ed esterne ben definite e superfici più spesse rispetto alla dimensione Δx di una cella della griglia MAC.

2.4 Incomprimibilità dei fluidi

Dopo una digressione prettamente geometrica della tesi, in questo paragrafo viene trattata la risoluzione delle equazioni (2.3a) e (2.3b) che permettono di rendere il fluido incomprimibile, garantendo le condizioni ai bordi.

2.4.1 Risolvere l'equazione della pressione: garantire incomprimibilità e condizioni ai bordi

L'algoritmo per la risoluzione di queste equazioni prevede di sottrarre al campo di velocità il gradiente della pressione, cioè

$$\vec{u}^{n+1} = \vec{u} - \Delta t \frac{1}{\rho} \nabla p, \tag{2.13}$$

in modo tale che il risultato soddisfi le condizioni di incomprimibilità del fluido

$$\nabla \cdot \vec{u}^{n+1} = 0, \tag{2.14}$$

le condizioni ai bordi sui confini solidi

$$\vec{u}^{n+1} \cdot \hat{n} = \vec{u}_{solido} \cdot \hat{n}$$

e quelle sulle superfici libere per le quali la pressione deve essere pari a zero

$$p = 0.$$

Prima di procedere, però, si deve adottare un modello adeguato per la rappresentazione del volume coinvolto nella simulazione, che solitamente risulta essere il *modello voxelized* basato sul concetto di *voxel*. Questo è un elemento discreto di un vettore di *elementi di volume* nei quali è possibile suddividere la rappresentazione tridimensionale di un oggetto. Il modello voxelized per la simulazione di fluidi prevede tre tipi di voxel: i voxel fluidi contenenti il fluido stesso, i voxel solidi contenenti un solido e i voxel vuoti contenenti aria non modellizzata. Grazie all'utilizzo di tale modello, è possibile identificare banalmente le superfici solide delle condizioni ai bordi come le facce tra un voxel solido ed uno liquido e, analogamente, le superfici libere come le facce comprese tra un voxel liquido e uno vuoto. Inoltre, l'utilizzo dei level set combinato con il modello voxelized semplifica notevolmente il controllo della posizione del centro di un voxel rispetto alla superficie del fluido, necessario per mettere in pratica la differenziazione sopra riportata.

Ricordando la discretizzazione dello spazio tramite le MAC grid e l'utilizzo delle differenze finite per il calcolo del gradiente dei valori memorizzati in essa, come la pressione, è possibile trasformare la (2.13) in

$$\begin{aligned} u_{i+1/2,j,k}^{n+1} &= u_{i+1/2,j,k} - \Delta t \frac{1}{\rho} \frac{p_{i+1,j,k} - p_{i,j,k}}{\Delta x}, \\ v_{i,j+1/2,k}^{n+1} &= v_{i,j+1/2,k} - \Delta t \frac{1}{\rho} \frac{p_{i,j+1,k} - p_{i,j,k}}{\Delta x}, \\ w_{i,j,k+1/2}^{n+1} &= w_{i,j,k+1/2} - \Delta t \frac{1}{\rho} \frac{p_{i,j,k+1} - p_{i,j,k}}{\Delta x}. \end{aligned} \tag{2.15}$$

Le (2.15) si applicano, però, solo alle componenti della velocità per le quali si ha un valore valido della pressione del fluido da entrambi i lati del voxel. Considerando il modello voxelized attualmente in uso, per evitare che i voxel fluidi contengono una pressione valida non nota che va calcolata, i voxel solidi contengono una pressione non valida mentre i voxel vuoti hanno una pressione valida con valore nullo. Conseguentemente, la (2.15) si applica solo per le componenti della velocità adiacenti ad almeno un voxel fluido, uno o nessun voxel vuoto e nessun voxel solido. Siccome i voxel vuoti possiedono pressione nulla, la condizione ai bordi per le superfici libere risulta facilmente applicabile e prende il nome di *condizione ai bordi Dirichlet* in quanto, quest'ultimo termine, indica che si sta specificando direttamente il valore di una quantità al bordo. Si hanno maggiori difficoltà nel caso delle condizioni ai bordi di superfici solide. La componente della velocità del fluido adiacente ad un voxel solido non viene aggiornata dalla (2.15) per quello specificato poc'anzi, di conseguenza è possibile settare il valore di tale

componente pari alla velocità del solido stesso. Tale processo, dovendo essere effettuato una volta completato l'aggiornamento di tutte le componenti della velocità valide, può essere considerato come un'estensione della (2.15), previa aggiunta di un valore *"fantasma"* della pressione all'interno del voxel solido. In tal caso, sia $p_{i,j,k}^{fantasma}$ tale valore, richiedere ad esempio che

$$u_{i+1/2,j,k}^{n+1} = u_{i+1/2,j,k}^{solido},$$

corrisponde alla formula (2.14) seguente

$$u_{i+1/2,j,k}^{n+1} = u_{i+1/2,j,k} - \Delta t \frac{1}{\rho} \frac{p_{i+1,j,k} - p_{i,j,k}^{fantasma}}{\Delta x},$$

nella misura in cui sia garantito che

$$p_{i,j,k}^{fantasma} = p_{i+1,j,k} - \frac{\rho \Delta x}{\Delta t}(u_{i+1/2,j,k} - u_{i+1/2,j,k}^{solido}). \tag{2.16}$$

Dalla (2.16) si ottiene

$$\frac{\Delta t}{\rho} \frac{p_{i+1,j,k} - p_{i,j,k}^{fantasma}}{\Delta x} = u_{i+1/2,j,k} - u_{i+1/2,j,k}^{solido}$$

da cui, sapendo che

$$\frac{\partial p}{\partial x} = \frac{p_{i+1,j,k} - p_{i,j,k}^{fantasma}}{\Delta x},$$

allora

$$\frac{\Delta t}{\rho} \frac{\partial p}{\partial x} = u_{i+1/2,j,k} - u_{i+1/2,j,k}^{solido}.$$

E' possibile ora sostituire, allo stesso modo, il valore del gradiente della pressione aggiornato alla condizione ai bordi per superfici solide, ottenendo

$$\frac{\Delta t}{\rho} \nabla p \cdot \hat{n} = \left(\vec{u} - \vec{u}^{solido}\right) \cdot \hat{n}.$$

Ricavando da quest'ultima la pressione, si ottiene che il valore della condizione ai bordi per superfici solide equivale alla derivata normale della pressione

$$\frac{\partial p}{\partial \hat{n}} = \nabla p \cdot \hat{n}$$

che viene tecnicamente definita *condizione ai bordi di Neumann*.

Prima di poter garantire l'incomprimibilità, è necessario capire come stimare la divergenza del campo di velocità. Si ricordi infatti che, per poter eseguire l'avvezione, è necessario che quest'ultimo sia a divergenza nulla, cioè

$$\nabla \cdot \vec{u} = 0.$$

Questo equivale a chiedere che nella griglia, utilizzando il metodo delle differenze finite, la stima della divergenza in ogni cella per il valore $\vec{u}^{n+1}$sia uguale a zero.

Essendo

$$\nabla \cdot \vec{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z},$$

utilizzando il metodo delle differenze finite si ottiene:

$$(\nabla \cdot \vec{u})_{i,j,k} \approx \frac{u_{i+1/2,j,k} - u_{i-1/2,j,k}}{\Delta x} + \frac{v_{i,j+1/2,k} - v_{i,j-1/2,k}}{\Delta x} + \frac{w_{i,j,k+1/2} - w_{i,j,k-1/2}}{\Delta x}. \quad (2.17)$$

Va specificato che la (2.17) viene utilizzata per calcolare la divergenza dei voxel fluidi. Un altro modo per valutarla è quello di usare il *metodo dei volumi finiti* il quale, però, sarà trattato in un paragrafo a parte in quanto estremamente utile nel caso in cui le superfici solide, interessate dalle condizioni ai bordi, siano curvilinee. Essendo ora in possesso di meccanismi per aggiornare le velocità nella griglia MAC tramite il gradiente di pressione, e per calcolare la divergenza in ogni cella, è necessario trovare il giusto valore di pressione tale per cui il nuovo campo $\vec{u}^{n+1}$ sia a divergenza nulla. Per fare ciò, si sostituisca la (2.15) nella (2.17) per ogni cella della griglia (*i*,*j*,*k*), esclusivamente di fluido, ottenendo:

$$\frac{u^{n+1}_{i+1/2,j,k} - u^{n+1}_{i-1/2,j,k}}{\Delta x} + \frac{v^{n+1}_{i,j+1/2,k} - v^{n+1}_{i,j-1/2,k}}{\Delta x} + \frac{w^{n+1}_{i,j,k+1/2} - u^{n+1}_{i,j,k-1/2}}{\Delta x} = 0, \quad (2.18)$$

da cui

$$\begin{aligned}\frac{1}{\Delta x}\Bigg[&\left(u_{i+1/2,j,k} - \Delta t \frac{1}{\rho}\frac{p_{i+1,j,k} - p_{i,j,k}}{\Delta x}\right) - \left(u_{i-1/2,j,k} - \Delta t \frac{1}{\rho}\frac{p_{i,j,k} - p_{i-1,j,k}}{\Delta x}\right) \\ &+ \left(v_{i,j+1/2,k} - \Delta t \frac{1}{\rho}\frac{p_{i,j+1,k} - p_{i,j,k}}{\Delta x}\right) \\ &- \left(v_{i,j-1/2,k} - \Delta t \frac{1}{\rho}\frac{p_{i,j,k} - p_{i,j-1,k}}{\Delta x}\right) \\ &+ \left(w_{i,j,k+1/2} - \Delta t \frac{1}{\rho}\frac{p_{i,j,k+1} - p_{i,j,k}}{\Delta x}\right) \\ &- \left(w_{i,j,k-1/2} - \Delta t \frac{1}{\rho}\frac{p_{i,j,k} - p_{i,j,k-1}}{\Delta x}\right)\Bigg] = 0\end{aligned}$$

ovvero

$$\begin{aligned}\frac{\Delta t}{\rho}&\left(\frac{6p_{i,j,k} - p_{i+1,j,k} - p_{i,j+1,k} - p_{i,j,k+1} - p_{i-,j,k} - p_{i,j-1,k} - p_{i,j,k-1}}{\Delta x^2}\right) \\ &= -\left(\frac{u_{i+\frac{1}{2},j,k} - u_{i-\frac{1}{2},j,k}}{\Delta x} + \frac{v_{i,j+\frac{1}{2},k} - v_{i,j-\frac{1}{2},k}}{\Delta x} + \frac{w_{i,j,k+\frac{1}{2}} - w_{i,j,k-\frac{1}{2}}}{\Delta x}\right)\end{aligned} \quad (2.19)$$

L'equazione (2.19) rappresenta l'approssimazione numerica del *problema di Poisson* $-\Delta t/\rho \nabla \cdot \nabla p = -\nabla \cdot \vec{u}$. Quest'ultima, calcolata per ogni cella di fluido della griglia,

genera un sistema di equazioni lineari, aventi per incognita la pressione, che verrà opportunamente risolto nel paragrafo successivo.

Va specificato infine che, se la cella di fluido per la quale si sta calcolando l'equazione (2.19) è sul bordo, allora nel caso la cella adiacente sia vuota, il termine della pressione corrispondente nell'equazione si annulla, mentre nel caso sia solida viene sostituito con il calcolo ottenuto dalle condizioni ai bordi per superfici solide, cioè dall'equazione (2.16).

2.4.2 Modified Incomplete Cholesky Conjugate Gradient, Level Zero

Nel paragrafo precedente è stato definito un sistema di equazioni lineari tramite il calcolo, per ogni cella di fluido (*i*,*j*,*k*) della griglia, della (2.19). E' possibile considerare tale sistema nella forma

$$Ap = b, \tag{2.20}$$

dove *A* è la matrice dei coefficienti, *p* un vettore di incognite della pressione e *b* un vettore contenente le divergenze negative per ogni cella di fluido della griglia. A livello implementativo è possibile memorizzare *p* e *b* in strutture a griglia tridimensionali mentre non risulta necessario memorizzare direttamente *A*.

Ogni riga di quest'ultima, infatti, è un'equazione che corrisponde ad una cella di fluido della griglia e, essendo le componenti in quella riga i coefficienti di tutte le pressioni non note, le uniche componenti diverse da zero saranno quelle della cella stessa e delle celle di fluido vicine. Conseguentemente, è possibile affermare che la matrice dei coefficienti *A* è *sparsa* e *simmetrica*. Queste proprietà permettono di memorizzare solo la metà dei valori non nulli presenti in essa. La matrice *A*, così definita, è stata oggetto di numerosi studi e prende il nome di *matrice Laplaciana a cinque o sette punti*. Per mantenere in memoria la suddetta matrice è possibile inserire, all'interno di una cella di fluido (*i*,*j*,*k*), solo la componete diagonale e le componenti relative alle celle vicine in direzione positiva, in quanto tutti gli altri valori possono essere trovati per simmetria.

La ricerca dell'adeguato valore di pressione per il quale il campo di velocità calcolato con la (2.15) è a divergenza nulla, è quindi, la risoluzione del sistema (2.20).

Uno dei metodi più efficienti per la risoluzione di tale sistema è il *Modified Incomplete Cholesky Conjugate Gradient Zero Level* o MICCG(0), il quale si rivela molto utile siccome la matrice dei coefficienti *A* è *simmetrica definita positiva* (SPD). Affermare ciò, significa che *A* è simmetrica e $q^T Aq > 0$ per ogni vettore q non nullo. Quest'ultima condizione non risulta sempre verificata, infatti solo nel caso in cui una cella fluido sia

racchiusa da celle solide, allora *A* risulterebbe una matrice singolare e, quindi, non SPD. Tale caso sarà accennato alla fine del capitolo in quanto di minore importanza.

Il metodo MICCG(0) vede alla base l'utilizzo del *metodo dei gradienti coniugati* o *Conjugate Gradient* (CG) per la risoluzione di SPD. CG è un metodo iterativo che, a partire da una possibile soluzione, ad ogni iterazione migliora quest'ultima fino a che non arriva ad un risultato abbastanza preciso, minimizzando una particolare misura dell'errore e garantendo la convergenza del metodo stesso. In aggiunta a tale vantaggio, CG utilizza operazioni facili da implementare anche in parallelo, quali moltiplicazione di una matrice per un vettore, moltiplicazione di vettori per scalari e prodotti scalari. Per evitare che il tempo di convergenza aumenti indefinitamente con l'aumentare della dimensione della matrice cioè della griglia, essendo proporzionali, si preferisce utilizzare una modifica di quest'ultimo definita *metodo dei gradienti coniugati precondizionato* o *Preconditioned Conjugate Gradient* (PCG).

E' necessario ora definire il precondizionatore e, per fare ciò, è possibile utilizzare un metodo della famiglia dei *Cholesky incompleti* o *Incomplete Cholesky* (IC) o IC(0), che prendono il nome banalmente a partire dalla *fattorizzazione di Cholesky*.

Il metodo IC(0) è un ottimo risultato comunemente utilizzato che, per una griglia di n celle di larghezza, converge in $O(n)$ iterazioni. E' tuttavia possibile ottenere una convergenza migliore pari a $O(n^{1/2})$ iterazioni, utilizzando un metodo modificato che prende il nome *Cholesky incompleto modificato* o *Modified Incomplete Cholesky* (MIC).

Prima di continuare la trattazione, è necessario soffermarsi su un aspetto non ancora introdotto, riguardante il metodo PCG cioè le condizioni che permettono di raggiungere la convergenza. Un buon meccanismo sarebbe quello di verificare la norma fra la soluzione corrente, ad una data iterazione, e la soluzione esatta che, tuttavia, non si conosce. E' necessario, quindi, introdurre il *vettore residuo* definito come

$$r_i = b - Ap_i.$$

Se la norma del vettore r_i è uguale a zero, cioè si è riusciti a soddisfare l'equazione $Ap = b$, oppure è sotto una certa tolleranza, le iterazioni si possono interrompere. Come norma è possibile prendere la norma infinito del residuo, mentre come tolleranza si è soliti usarne una adimensionale, $\sim 10^{-6}$, richiedendo quindi che il metodo si interrompa se:

$$\|r\| = \|b - Ap\| \leq tol\|b\|.$$

Inoltre, è importante fornire un numero massimo di iterazioni, ad esempio 200, oltre le quali converrebbe fermare il metodo, in quanto, potrebbero verificarsi degli errori derivanti dall'utilizzo *dell'aritmetica floating point*.

Infine, essendo PCG la parte più onerosa, a livello computazionale, del programma per la simulazione di un fluido, è consigliabile l'utilizzo di API specifiche quali *BLAS,* che forniscono funzioni ottimizzate per lo svolgimento di operazioni vettoriali anche in parallelo.

Tale aspetto di parallelizzazione, legato alla buona scelta di scomposizione del dominio, risulta necessario per un'implementazione di PCG che permetta di ridurre i costi e i tempi computazionali. Scomporre il dominio in molteplici sotto-domini, ognuno dei quali rappresentante un sistema lineare più piccolo, permette di svolgere la risoluzione di quest'ultimi in modo parallelo, ottimizzando i tempi. Una scomposizione efficace del dominio può essere effettuata con una variante di ciò che è conosciuto con *Additive Overlapping Schwarz*. La griglia, ovvero il dominio, viene partizionata sulla base dei seguenti criteri: il numero di sotto-griglie generate deve essere almeno pari al numero di *thread* che si vogliono utilizzare; le sotto-griglie, nel loro insieme, devono coprire l'intera area, volume della griglia; le sotto-griglie devo avere qualche cella della griglia in comune. Sia *z* il risultato dell'applicazione del precondizionatore *r*, prima di calcolarlo, è necessario risolvere la parte di equazione $Az = r$ per ogni sotto-griglia indipendentemente. Siccome tale operazione prende parte in un ciclo PCG esterno, approssimazioni non simmetriche o non lineari comporterebbero problemi di convergenza, tuttavia, la risoluzione separata sopracitata permette una veloce e facile parallelizzazione. Una volta che tutte le sotto-griglie hanno un loro valore di *z* relativo alla sotto-griglia stessa, sarà necessario sommare le soluzioni intermedie per ottenere una stima globale di *z* che sarà restituita al ciclo di PCG esterno. Per ottenere, poi, un precondizionatore dalle alte prestazioni è possibile fare uso del metodo *Discretely-Discontinuous Galerkin* (DDG), introdotto da Edwards e Bridson [EB15].

Con questo paragrafo è finito lo studio delle equazioni (2.1), (2.3a) e (2.3b) che permettono la simulazione del fluido e garantiscono, una volta risolte, l'incomprimibilità dello stesso e il rispetto delle condizioni ai bordi. Prima di procedere con la trattazione degli ultimi metodi, necessari per poter effettuare la rappresentazione di un fluido quale l'acqua, si vuole studiare come tecniche e modellazioni viste nei paragrafi precedenti

vengano modificate nel caso le superfici, sulle quali si applicano le condizioni ai bordi, siano curve.

2.4.3 Metodo dei volumi finiti per bordi curvi

Come accennato nel paragrafo precedente, spesso le superfici, sulle quali si vogliono imporre le condizioni ai bordi, non coincidono con le facce delle celle della griglia. Questo, abbinato al modello voxelized, può generare problemi grafici in fase di simulazione. Da un punto di vista strettamente matematico, nel caso sopra citato, il problema nasce dal fatto che la componente normale della velocità non risulta convenientemente memorizzata nella griglia e quindi l'interpolazione non può essere effettuata correttamente. Una soluzione si trova nell'uso di *mesh tetraedriche destrutturate* con bordi allineati alle superfici del solido. Tale approccio introduce, però, problemi di performance durante le operazione di costruzione delle mesh e di lavoro con le stesse, conseguentemente, si preferisce adottare una tecnica di discretizzazione differente rispetto al modello voxelized precedentemente trattato pur continuando a lavorare con griglie cartesiane.

Alternativamente al modello voxelized è possibile utilizzare, infatti, il *metodo dei volumi finiti* il quale, lavorando sulle condizioni ai bordi di superfici solide, va a modificare, rispetto a quanto finora detto nei paragrafi precedenti, solo la condizione di divergenza. Si consideri la forma integrale della condizione di incomprimibilità (1.4)

$$\iint_{\partial\Omega} \vec{u} \cdot \hat{n} = 0,$$

dove $\partial\Omega$ è la superficie di bordo di un volume di controllo Ω.

Nello specifico, per la parte interna al flusso, si considera ogni cella della griglia come un volume di controllo e si approssima l'integrale del bordo, su ogni faccia, come l'area Δx^2della faccia moltiplicata per la componente normale della velocità memorizzata nel suo centro. Sul bordo, se il solido taglia la faccia di una cella della griglia oppure un volume di controllo, allora l'integrale sopra riportato possiede due contributi: la velocità del fluido relativamente alla parte di quest'ultimo della faccia della cella e, analogamente, la velocità del solido per la rimanente parte della faccia della cella. Conseguentemente, l'approssimazione totale sarà data da quella poc'anzi specificata rispetto ai due contributi divisa per Δx^2, cioè

$$-F_{i-1/2,j,k}u_{i-1/2,j,k} + F_{i+1/2,j,k}u_{i+1/2,j,k}$$
$$-F_{i,j-1/2,k}u_{i,j-1/2,k} + F_{i,j+1/2,k}u_{i,j+1/2,k}$$

$$-F_{i,j,k-1/2}u_{i,j,k-1/2} + F_{i,j,k+1/2}u_{i,j,k+1/2}$$
$$-(1 - F_{i-1/2,j,k})u^{solido}_{i-1/2,j,k} + (1 - F_{i+1/2,j,k})u^{solido}_{i+1/2,j,k}$$
$$-(1 - F_{i,j-1/2,k})u^{solido}_{i,j-1/2,k} + (1 - F_{i,j+1/2,k})u^{solido}_{i,j+1/2,k}$$
$$-\left(1 - F_{i,j,k-1/2}\right)u^{solido}_{i,j,k-1/2} - \left(1 - F_{i,j,k+1/2}\right)u^{solido}_{i,j,k+1/2} = 0, \qquad (2.22)$$

dove i termini *F* indicano le porzioni di area delle facce del fluido, nell'intervallo di valori [0,1] con zero per una cella totalmente di fluido e uno per una cella totalmente di solido. Applicando la formula del gradiente della pressione alla (2.22), ovvero la nuova condizione di divergenza dei volumi finiti, si ottiene un *sistema lineare simmetrico definito semi-positivo*.

Tale modello di discretizzazione assume che la velocità del solido abbia divergenza nulla e, a differenza del modello voxelized, che le pressioni memorizzate al centro di celle che si trovano all'interno di solidi non vengono ignorate. Infine, il modello dei volumi finiti non permette la gestione di solidi molto sottili.

Rimane ora da capire come calcolare le frazioni delle facce del fluido *F*. Se la geometria del fluido è fornita sotto forma di mesh polinomiali, allora basta risolvere l'intersezione delle facce della mesh con le celle della griglia. Un'ulteriore semplificazione avviene se la geometria è fornita sotto forma di level set campionato agli angoli dei voxel. Sia ϕ(x) la funzione level set che definisce implicitamente la geometria del solido e siano ϕ_0, ϕ_1 e ϕ_2 i valori campionati di quest'ultima nei vertici 0,1,2 di un triangolo tali che $\phi_0 < \phi_1 < \phi_2$. Sia poi definita per $\phi(x) < 0$ la regione di solido, per $\phi(x) > 0$ la regione di fluido e per $\phi(x) = 0$ la superficie del solido, allora se $\phi_0 > 0$ è possibile affermare che la superficie del solido non interseca il triangolo, cioè quest'ultimo è solido, analogamente se $\phi_2 < 0$ il triangolo è completamente fluido.

Nel caso in cui $\phi_0 \leq \phi_1 < 0 < \phi_2$ allora solo il vertice 2 è fluido, di conseguenza $\phi(x)$ viene interpolata linearmente a zero sul bordo 0-2 per il seguente valore

$$\theta_{02} = \frac{\phi_2}{\phi_2 - \phi_0}.$$

Analogamente, la porzione del bordo 1-2 interna al fluido è

$$\theta_{12} = \frac{\phi_2}{\phi_2 - \phi_1}$$

e la porzione *F*, dell'intera area del triangolo, è data dal prodotto delle due frazioni sopra, cioè

$$F_{triangolo} = \theta_{02}\theta_{12}.$$

Infine, nel caso $\phi_0 < 0 < \phi_1 \leq \phi_2$, il vertice 0 sarà l'unico presente nel solido e la porzione di area sarà calcolata come l'opposto della soprastante, cioè

$$F_{triangolo} = 1 - \left(\frac{\phi_0}{\phi_0 - \phi_1}\right)\left(\frac{\phi_0}{\phi_0 - \phi_2}\right).$$

In questa sede si vuole mostrare la tecnica per calcolare una porzione di faccia partendo da un triangolo, tuttavia è possibile trovare il valore di *F* per qualsiasi geometria, suddividendola in triangoli e utilizzando le formule sopra. E' importante, inoltre, sottolineare che tale metodo tende a produrre gravi errori nel calcolo della pressione per celle nelle quali sono presenti porzioni di facce molto piccole, perciò, per valori inferiori di una certa soglia, è bene approssimare a zero quest'ultime [BBB07].

2.4.4 Condizione di compatibilità

La *condizione di compatibilità* è necessaria nella misura in cui una regione di fluido sia racchiusa da un solido senza la presenza di superfici libere. Tale situazione, introdotta nei paragrafi precedenti, porterebbe all'assenza di soluzione per la PDE relativa alla pressione. La condizione di compatibilità

$$\iiint_{\Omega} \nabla \cdot \vec{u} = \iint_{\partial\Omega} \vec{u} \cdot \hat{n},$$

diretto risultato del teorema della divergenza, assicurerebbe il contrario. Per ottenere ciò si può considerare il problema come fosse un semplice problema di algebra lineare [GSLF05], tuttavia, essendo la casistica di tale situazione notevolmente rara e fuori dallo scopo di tale tesi, se ne tralascia lo studio.

Capitolo 3

3 Simulazione di fluidi

Nel capitolo precedente, sono state trattate le equazioni di avvezione e di incomprimibilità ovvero il cuore della simulazione dei fluidi. La loro modellizzazione, discretizzazione e risoluzione permette, come già detto, di descrivere completamente il comportamento di un fluido. In questo capitolo, si vuole proporre un metodo alternativo all'approccio semi-lagrangiano, utilizzato per l'avvezione, che permetta di risolvere i problemi introdotti da quest'ultima quali, ad esempio, l'elevata diffusione numerica causata dall'interpolazione.

3.1 Metodo misto per la riduzione della diffusione numerica

Come accennato poc'anzi, l'interpolazione, utilizzata durante l'avvezione, introduce un'elevata diffusione numerica. L'uso di un interpolazione polinomiale cubica, rispetto ad una semplice interpolazione lineare, può ridurre tale diffusione che, però, rischia comunque di causare problemi di risoluzione grafica. Un modo per vedere tale problema è quello di analizzare gli step effettuati da tutti i metodi euleriani, anche i cosiddetti semi-lagrangiani, durante l'avvezione. Questi procedono campionando il campo di velocità sulla griglia, ricostruiscono il campo come una funzione continua a partire dai valori campionati, si effettuano l'avvezione del campo ricostruito ed infine campionano nuovamente il campo sulla griglia. Guardando tale sequenza di passi, è possibile notare il problema fondamentale dei campi di velocità incomprimibili, infatti, pur preservando il volume del fluido, ad ogni iterazione potrebbero essere allungati, stirati oppure ristretti su uno o più assi.

Nel caso del re-campionamento di un campo di velocità dopo uno stiramento dello stesso, non si ha perdita di informazioni, mentre è possibile che ciò non avvenga a seguito di un restringimento. In quest'ultimo caso, le suddette informazioni, potrebbero essere perse o iniziare a comportarsi come artefatti spuri a bassa frequenza cioè, a livello di rendering, disturbi dell'immagine. Come già accennato nei capitoli precedenti, tali disturbi potrebbero essere intesi come effetti fisici del fluido, ad esempio la viscosità, oppure risultare mascherati dalla presenza di processi di diffusione molecolare. Quest'ultimo caso, nel quale un fenomeno fisico limita il campionamento delle informazioni ad alta frequenza disperdendole, suggerisce l'uso della *simulazione numerica diretta* (DNS). La

DNS risulta essere, però, molto costosa per quanto riguarda la maggior parte dei lavori di grafica a causa del fatto che, le risorse di calcolo necessarie alla loro risoluzione, crescono con il numero di Reynolds (quasi con Re^3) e che tale numero può avere valori dell'ordine di 10^6-10^9. Anche l'utilizzo di *griglie adattive*, ovvero griglie in grado di modificare la propria risoluzione sulla base della densità delle informazioni da campionare, presentano problemi di performance che ne limitano l'uso.

3.1.1 Avvezione particellare

L'idea alla base del metodo misto, è quella di memorizzare il campo di velocità all'interno delle particelle di un sistema particellare. Ricollegandosi all'approccio lagrangiano e a come è stata definita l'avvezione, la (2.1) ci permette di dire che, qualsiasi sia la modifica apportata dal flusso alla distribuzione di particelle, le informazioni memorizzate in esse non cambiano. Di seguito si riporta, quindi, la tecnica necessaria per sostituire l'avvezione semi-lagrangiana, trattata nel paragrafo 2.2, con *l'avvezione particellare*. Per risolvere l'ODE valgono le considerazione effettuate nel capitolo precedente, cioè è necessario utilizzare, per evitare errori numerici, un metodo RK di ordine superiore al secondo oppure il seguente schema proposto da Ralston [RAL62]:

$$\vec{k}_1 = \vec{u}(\vec{x}_n),$$
$$\vec{k}_2 = \vec{u}\left(\vec{x}_n + \frac{1}{2}\Delta t\vec{k}_1\right),$$
$$\vec{k}_3 = \vec{u}\left(\vec{x}_n + \frac{3}{4}\Delta t\vec{k}_2\right),$$
$$\vec{x}_{n+1} = \vec{x}_n + \frac{2}{9}\Delta t\vec{k}_1 + \frac{3}{9}\Delta t\vec{k}_2 + \frac{4}{9}\Delta t\vec{k}_3.$$

Come per l'avvezione semi-lagrangiana è possibile utilizzare, per controllare in modo più preciso l'errore, più sotto-iterazioni.

Uno dei problemi introdotti dalla risoluzione della (2.1), maggiormente trattati nel capitolo precedente, è relativo alle condizioni ai bordi. Nel caso dell'avvezione particellare, grazie all'utilizzo dei level set, è banale capire se una particella risulta oltre i confini solidi. Conseguentemente, se questa, per errori numerici o altro, si trova all'esterno del volume del fluido, è possibile modificare la sua posizione per rimetterla all'interno di quest'ultimo, ad esempio nel punto più vicino alla superficie, oppure eliminarla.

Ora, che si è introdotta la tecnica di avvezione particellare e le differenze con quella precedentemente esposta, è possibile affermare che, se la simulazione prevede la presenza

di diffusione fisica importante ad alta intensità, allora è consigliato l'utilizzo del metodo euleriano (avvezione semi-lagrangiana). Nel caso in cui ciò non avvenga o sia necessario tracciare all'interno del campo altre proprietà che possiedo un riscontro grafico, quali concentrazione di fumo, schiuma etc., è consigliato l'uso dell'avvezione particellare.

Passiamo ora alla trattazione del metodo alla base dell'avvezione particellare. Negli anni cinquanta, nei laboratori nazionali di Los Alamos, fu studiato per la prima volta il metodo *particle-in-cell* (PIC). Quest'ultimo, introdotto solamente nel 1963 da Harlow [HAR63], semplicemente memorizzando i vettori delle velocità nelle particelle, permette di effettuare l'avvezione particellare, introdotta poc'anzi, al posto dell'avvezione semi-lagrangiana delle velocità. Il metodo PIC, rispetto alla risoluzione dell'equazione di avvezione (2.1), richiede, però, che vengano trattate anche le interazioni particella-particella una volta completata l'operazione. Prima di procedere con la logica del metodo PIC, è necessario trattare come le particelle vengono emesse all'interno del dominio della simulazione e come le quantità memorizzate al loro interno vengono trasferite nella griglia.

Caratteristica essenziale dei metodi per l'emissione di particelle è l'indipendenza della quantità di particelle emessa dall'intervallo di tempo scelto e dalla dimensione delle celle della griglia. Utilizzare un approccio che non garantisce ciò, porterebbe ad un comportamento totalmente errato dell'emissione delle particelle. Esistono due approcci principali per effettuare, correttamente, l'emissione di quest'ultime: emissione spazialmente uniforme e emissione ad aree nettamente definite.

Il primo approccio viene utilizzato quando è presente una regione nella quale la densità di emissione è costante e si riduce a zero ai bordi. In questo caso, l'emissione di W particelle avviene in posizioni casuali per ogni cella con densità diversa da zero ed è possibile specificare un tasso di emissione e, conseguentemente, di creazione delle particelle come dW/dt espresso in particelle per voxel al secondo. Così facendo, verranno create ed emesse $n = \Delta t dW/dt$ particelle in maniera continuativa e uniforme. Infine, è possibile agire sul tempo di creazione, assegnandone una casuale, per distribuire l'emissione delle particelle in modo uniforme ad ogni intervallo di tempo. Il secondo approccio è utile nelle situazioni in cui è presente una regione, definita ad esempio da un level set, nella quale la densità di emissione è diversa da zero e cala drasticamente annullandosi fuori da essa. In tal caso, la particella viene creata in una posizione casuale e se questa appartiene alla regione definita dal level set, viene emessa.

Prima di passare al metodo PIC, si vuole mostrare come sia possibile trasferire proprietà memorizzate dalle particelle alla griglia. Tale meccanismo può essere ovviamente utilizzato per qualsiasi quantità. Si definisca $\vec{x}_p$ la posizione iniziale della particella p e q_P il valore della quantità q. E' possibile impostare tale valore quando le particelle vengono create, oppure assegnare a tutte un valore di default. La concentrazione della quantità q nella cella (i,j,k) della griglia sarà

$$q_{i,j,k} = \sum_p q_p \frac{k(\vec{x}_p - \vec{x}_{i,j,k})}{W},$$

dove k è una funzione che assegna un peso alle particelle vicine e W il peso per normalizzare il valore della particella. Nel caso il supporto della funzione k sia inferiore alla dimensione di una cella, è necessario normalizzarla, ad esempio, con una B-spline quadratica. Si può stimare invece il valore totale della quantità q, nel volume della simulazione, utilizzando le particelle come

$$Q_{totale} \approx \sum_p q_p V,$$

oppure tramite la griglia come

$$Q_{totale} \approx \sum_{i,j,k} q_{i,j,k} \Delta x^3.$$

Per stimare un valore adeguato per W, invece, basta considerare il numero medio di particelle per cella della griglia, cioè

$$W = \frac{\Delta x^3}{V}.$$

Avendo queste conoscenze, è possibile completare la spiegazione del metodo PIC.

3.1.2 Il metodo PIC-FLIP

Questo inizia con un insieme di particelle che campionano tutta la regione di fluido, in cui sono memorizzate tutte le quantità necessarie per descrivere quest'ultimo. Ad ogni iterazione, le quantità delle quali si vuole fare l'avvezione, quali la velocità, vengono trasferite come indicato sulla griglia per poi essere interpolate da questa di nuovo alle particelle. Fatto ciò, viene eseguita l'avvezione particellare sul campo di velocità della griglia assicurandosi, per una maggiore accuratezza, che il peso W sopracitato sia calcolato per ogni cella della stessa. E' infine necessario, ottenendo dal trasferimento della quantità dalla particella alla griglia solo valori vicini alla particella stessa, estrapolare il risultato a quelle vicine, come spiegato nei capitoli precedenti.

Il metodo PIC è connotato, però, come l'avvezione semi-lagrangiana, da una grande diffusione numerica. Per risolvere questo problema Brackbill e Ruppel [BR86]

introdussero una modifica al metodo PIC chiamato *fluid implicit particle* (FLIP). Il metodo FLIP, invece di interpolare le quantità dalla griglia alle particelle, interpola la variazione della quantità aggiungendola al precedente valore della particella stessa. FLIP, pur introducendo un piccolo *smoothing* e, a volte, un po' di perturbazione nei valori, risulta teoricamente privo di diffusione numerica, il che lo rende, nel caso le particelle fossero utilizzate anche per altre quantità oltre la velocità, il metodo migliore per effettuare l'avvezione. Nel caso si vogliano ottenere risultati migliori, è possibile unire il metodo PIC, con il metodo FLIP, che è teoricamente privo di diffusione numerica per azzerare le perturbazioni introdotte dal primo. Scelto un parametro di regolarizzazione α, basta impostare la velocità delle particelle a

$$\vec{u}_p^{nuovo} = \alpha\, interp\left(\vec{u}_{griglia}^{nuovo}, \vec{x}_p\right) + (1-\alpha)[\vec{u}_p^{vecchio} + interp\left(\Delta\vec{u}_{griglia}, \vec{x}_p\right)].$$

Se α = 0 allora l'aggiornamento è dato da FLIP, altrimenti se α = 1 è dato da PIC.

E' infine possibile legare α al valore della viscosità cinematica ν, ottenendo

$$\alpha = \frac{6\Delta t \nu}{\Delta x^2}.$$

Ciò risulterà molto utile nei capitoli successivi dove verrà trattato questo argomento.

I metodi PIC e FLIP risultano essenziali per una rappresentazione priva, almeno per quanto riguarda il secondo, di diffusione numerica che è un problema intrinseco all'avvezione semi-lagrangiana.

3.2 L'acqua

Si hanno ora tutti gli elementi per rappresentare un fluido come l'acqua al calcolatore. In questo paragrafo verranno trattati, quindi, gli ultimi aspetti geometrici e di modellazione e principalmente grafici per ottenere lo scopo.

Si consideri l'acqua come fluido, avente condizioni ai bordi come specificato all'inizio della trattazione e sia definita la superficie libera aria-acqua *interfaccia*. L'unico problema di modellizzazione del dominio e di geometria non trattato, riguarda l'identificazione e l'aggiornamento delle celle-voxel fluidi e di quelle vuote, cioè di aria, e come queste si scambiano quando l'acqua si muove a seguito dell'avvezione del campo di velocità. Per risolvere questo problema si abbina al modello voxelized, l'utilizzo delle *marker particles,* introdotte da Harlow e Welch nel 1965 [HW65]. Risulta necessario adottare tale approccio in quanto, l'uso di level set del quale si esegue l'avvezione e un'interpolazione cubica, mostra artefatti grafici legati all'impossibilità di trattare superfici più sottili di una cella della griglia, producendo la scomparsa, ad esempio, delle

particelle di acqua che si staccano dalla superficie. L'idea alla base delle marker particles è quella di usare queste speciali particelle, dopo lo step dell'avvezione, per segnare sulla griglia quali sono di fluido. Il numero di marker particles consigliato è di otto per ognuna delle celle in 3D. L'avvezione può essere effettuata sia con il metodo semi-lagrangiano, sia con il metodo PIC o FLIP introdotti nel paragrafo precedente. Il problema grafico sopra citato è relativo, invece, alla rappresentazione grafica dell'interfaccia dell'acqua in quanto, il rendering diretto dei voxel fluidi, non fornirebbe risultati soddisfacenti. Per un interesse personale si vogliono mostrare diverse tecniche possibili la cui trattazione dettagliata si può trovare negli articoli citati. Un primo metodo, *blobbies,* introdotto da Blinn permette di costruire una superficie liscia implicita avvolta attorno alle particelle [BLI82]. Tale metodo genera la superficie *F*, date le posizioni delle particelle $\{\vec{x}_i\}$, come

$$F(\vec{x}) = \sum_i k\left(\frac{\|\vec{x}-\vec{x}_i\|}{h}\right),$$

dove *k* è una funzione kernel e *h* un parametro scelto dall'utente per essere l'estensione di ciascuna particella. La funzione *k* può essere definita come

$$k(s) = \begin{cases} (1-s^2)^3 & : s < 1 \\ 0 & : s \geq 1 \end{cases}$$

L'utilizzo dei blobbies, però, produce artefatti grafici eliminabili sacrificando caratteristiche che in una simulazione potrebbe essere interessante mostrare.

Un miglioramento ai blobbies è stato apportato da Zhu e Bridson [ZB05] i quali calcolano la funzione della superficie implicita come

$$\phi(\vec{x}) = \|\vec{x} - \bar{X}\| - \bar{r},$$

dove $\bar{X}$ è una media pesata delle posizioni di tutte le particelle vicine

$$\bar{X} = \frac{\sum_i k\left(\frac{\|\vec{x}-\vec{x}_i\|}{h}\right)\vec{x}_i}{\sum_i k\left(\frac{\|\vec{x}-\vec{x}_i\|}{h}\right)},$$

e $\bar{r}$ è una media pesata dei raggi delle particelle vicine

$$\bar{r} = \frac{\sum_i k\left(\frac{\|\vec{x}-\vec{x}_i\|}{h}\right)\vec{r}_i}{\sum_i k\left(\frac{\|\vec{x}-\vec{x}_i\|}{h}\right)}.$$

Anche tale metodo, però, produce in certe situazioni artefatti grafici, risultanti in buchi sulla superficie. Vi sono ulteriori metodi per la creazione geometrica dell'interfaccia dell'acqua tuttavia, negli ultimi anni si è iniziato ad utilizzarne uno che evita l'uso dei level set e crea la superficie a partire direttamente dalle mesh triangolari introdotte nei capitoli precedenti. Utilizzando tale tecnica è possibile gestire l'avvezione, invece che di particelle, dei vertici delle mesh e, ricalcolando quest'ultime ad ogni iterazione con

un'operazione di *remesh*, è possibile gestire i problemi grafici derivanti dalla compressione e dilatazione delle stesse. Prima di mostrare l'algoritmo grazie al quale è possibile fare la simulazione dell'acqua, è necessario fare una modifica alla parte di metodo dedicata alla risoluzione dell'equazione della pressione. Il problema in questione nasce dal fatto che tale parte di metodo tratta la condizione al bordo per superfici libere, imponendo $p = 0$, sulla base del modello voxelized. Conseguentemente, anche se si è riusciti a renderizzare, con le tecniche sopra riportate, una superficie realistica di acqua con parti di cella segnate come tale ed altre segnate come aria, questo le considera come voxel. Ciò comporta la presenza, a livello grafico, di artefatti derivanti dalla natura voxelized del modello. E' quindi necessario modificare il calcolo del gradiente della pressione, necessario per l'aggiornamento delle velocità, nelle celle di superficie. Il metodo più diffuso per risolvere il problema sopra definito, denominato *metodo del fluido fantasma*, è stato introdotto da Gibou nel 2002.

Il metodo del fluido fantasma per le celle interne al fluido, considerando ad esempio $u_{i+1/2,j,k}^{n+1}$ la velocità aggiornata di una di esse, è la (2.15)

$$u_{i+1/2,j,k}^{n+1} = u_{i+1/2,j,k} - \frac{\Delta t}{\rho_{i+1/2,j,k}} \frac{p_{i+1,j,k} - p_{i,j,k}}{\Delta x}. \tag{3.1}$$

Sia la cella (i,j,k) nell'acqua, cioè $\phi_{i,j,k} \leq 0$ e sia la cella (i+1,j,k) nell'aria, cioè $\phi_{i+1,j,k} > 0$ allora il metodo fino ad ora adottato e trattato nei capitoli precedenti porrebbe $p_{i+1,j,k} = 0$. Tuttavia, analogamente al metodo dei volumi finiti, sarebbe maggiormente corretto affermare che $p = 0$ sull'interfaccia dell'acqua che taglia il bordo (i,j,k), (i+1,j,k). Di conseguenza, per trovare il punto per la quale quest'ultima passa, è necessario interpolare linearmente tra (i,j,k) e (i+1,j,k) ottenendo

$$(i + \theta\Delta x, j, k)$$

dove

$$\theta = \frac{\phi_{i,j,k}}{\phi_{i,j,k} - \phi_{i+1,j,k}}. \tag{3.2}$$

Si interpoli ora il valore reale della pressione $p_{i,j,k}$ con il valore fantasma $p_{i+1,j,k}^{fantasma}$ e si ponga il risultato uguale a zero all'interfaccia, ottenendo

$$(1 - \theta)p_{i,j,k} + \theta p_{i+1,j,k}^{fantasma} = 0$$

da cui, sostituendo la (3.2)

$$p_{i+1,j,k}^{fantasma} = -\frac{1-\theta}{\theta} p_{i,j,k} = \frac{\phi_{i+1,j,k}}{\phi_{i,j,k}} p_{i,j,k} .$$

Sostituendo questa alla (3.1) o (2.15), si ottiene

$$u_{i+\frac{1}{2},j,k}^{n+1} = u_{i+\frac{1}{2},j,k} - \frac{\Delta t}{\rho}\frac{\frac{\phi_{i+1,j,k}}{\phi_{i,j,k}}p_{i,j,k} - p_{i,j,k}}{\Delta x} = u_{i+\frac{1}{2},j,k} - \frac{\Delta t}{\rho}\frac{\phi_{i+1,j,k} - \phi_{i,j,k}}{\phi_{i,j,k}}\frac{p_{i,j,k}}{\Delta x}.$$

A livello matematico, le modifiche appena fatte comportano l'aumento della dimensione della diagonale, che semplifica la sua risoluzione e rende il tutto ben condizionato. Per evitare perturbazione numeriche è necessario, però, porre un limite massimo di almeno 10^3 al valore di $(\phi_{i+1,j,k} - \phi_{i,j,k})/\phi_{i,j,k}$.

Prima di concludere è necessario fare alcune considerazioni finali. L'acqua, di per se, tende a rilasciare gocce dalla superficie. Questo effetto viene gestito automaticamente dall'uso congiunto dei level set e delle marker particles. Ciò nasce, però, dagli errori numerici intrinseci ai metodi sopra citati, più precisamente, dal campionamento del liquido su un insieme finito di punti della griglia o di particelle. Un altro aspetto che, invece, non viene gestito è la separazione dalle superfici solide dell'acqua. Quando una goccia o uno spruzzo di acqua colpisce una parete solida, infatti, per la condizione ai bordi per superfici solide ovvero la condizione no-stick, questa non si stacca più. Tale condizione è però necessario, come detto nei capitoli precedenti, per evitare che gocce, particelle, etc. attraversino la superficie solida. A quest'ultimo problema è stata trovata una soluzione solo nel 2012, quando Chentanez e Müller hanno studiato l'utilizzo delle multi-griglie [CMF12].

In conclusione, si riporta l'algoritmo per la simulazione dell'acqua con l'utilizzo di tutte le tecniche mostrate nel corso di questo e dei precedenti capitoli:

1. Si riempie un volume, definito tramite un level set ad esempio, con particelle di acqua.
2. A partire dalle particelle, si costruisca il level set per il liquido.
3. Si trasferisca la velocità dalle particelle alla griglia, si estrapolano i valori conosciuti nella griglia ad almeno una cella attorno al fluido ottenendo un primo campo di velocità $\vec{u}^*$.
4. Si aggiungano le forze del corpo, quali la forza gravitazionale, al campo di velocità.
5. Si costruisca il level set per il solido e il campo di velocità per quest'ultimo.
6. Si risolva l'equazione della pressione e si applichi a $\vec{u}^*$per ottenere un campo di velocità a divergenza nulla $\vec{u}^{n+1}$che rispetta le condizioni ai bordi.
7. Si aggiornino le velocità delle particelle, utilizzando FLIP o PIC.

8. Fare l'avvezione delle particelle attraverso il campo di velocità a divergenza nulla $\vec{u}^{n+1}$.

Capitolo 4

4 Algoritmi avanzati per la simulazione di fluidi

Questo capitolo si focalizza sullo studio della teoria e degli algoritmi alla base della simulazione di fluidi differenti per proprietà dall'acqua oppure ambienti composti da quest'ultima, quali oceani e acque basse. Verranno quindi introdotti i concetti di viscosità e di turbolenza, necessari per una simulazione avanzata e la rappresentazione di proprietà e fenomeni tipici presenti in natura.

4.1 Viscosità

Finora la trattazione si è concentrata sui fluidi poco viscosi e non viscosi. Per simulare i primi non è necessario complicare i metodi risolutivi proposti con tecniche dedicate alla gestione della viscosità, come quelli mostrati di seguito. Infatti, i metodi definiti nei capitoli precedenti introducono perturbazioni numeriche che, a livello grafico, possono essere interpretate come l'effetto fisico della viscosità. Per i fluidi non viscosi, è possibile eliminare dalla (1.1) i termini relativi alla viscosità ottenendo così le *equazioni di Eulero*. Per simulare fluidi altamente viscosi, invece, è necessaria una modifica ai metodi già introdotti e l'introduzione di un concetto finora non trattato: lo *stress* o *sforzo*.

4.1.1 Sforzi, tensori e Teorema degli sforzi di Cauchy

Si consideri il modello del continuo, già accennato all'inizio della trattazione, secondo cui la materia è un campo continuo formato da particelle la cui dimensione e vicinanza tendono infinitamente a zero. Le forze che agiscono su tale campo possono applicarsi all'intero volume, oppure localmente come la pressione. In questo secondo caso è intuitiva la definizione delle stesse come *forze di contatto locali.* Per capire la viscosità, è necessario studiare tali forze e lo sforzo da esse prodotte.

Prima di procedere è doveroso fare una digressione sull'argomento dei *tensori,* fondamentali per il proseguo della trattazione. Un tensore è un vettore multidimensionale di valori numerici che descrive le relazioni lineari che intercorrono tra vettori geometrici, scalari o altri tensori. Esso possiede un ordine, il quale specifica il numero di dimensioni del vettore. Un tensore di primo ordine, ad esempio, è un vettore di una dimensione;

analogamente, un tensore di secondo ordine è un vettore di due dimensioni, cioè una matrice. Esprimendo relazioni tra vettori, il tensore deve essere indipendente da particolari scelte di sistemi di coordinate. Ciò gli conferisce particolare importanza perché permette di legare il vettore forza di contatto locale ad un vettore direzione, di lunghezza unitaria, perpendicolare alla superficie sulla quale si vuole calcolare la forza.

Si definisce *stato degli sforzi interni* di un corpo in un punto $\vec{x}$ l'insieme di tutti i vettori $\vec{t}^{(\hat{n})}$ del campo vettoriale $\vec{t}$, generato dalle forze di contatto locale, associati a tutti i piani passanti per quel punto. Il *teorema degli sforzi di Cauchy* [IRG08] permette di calcolare, noti i vettori forza di contatto locale su tre piani reciprocamente perpendicolari, il vettore forza su qualsiasi altro piano passante per quel punto, tramite equazioni di trasformazioni di coordinate. Il teorema sopracitato afferma che esiste un campo di tensori di secondo ordine σ chiamato *tensore degli sforzi di Cauchy*, indipendente da $\hat{n}$, tale per cui $\vec{t}^{(\hat{n})}$ è una funzione lineare di $\hat{n}$, cioè dato il punto $\vec{x}$ allora

$$\vec{t}(\vec{x}, \hat{n}) = \sigma(\vec{x})\hat{n},$$

oppure, nella forma comprendente i campi vettoriali, come

$$\vec{t} = \sigma\hat{n}, \tag{4.1}$$

Si consideri un fluido di volume **Ω**, la forza totale di contatto su quest'ultimo è data da

$$\vec{F} = \iint_{\partial\Omega} \vec{t}(\vec{x}, \hat{n}). \tag{4.2}$$

Siccome un fluido non viscoso esercita delle forze solo lungo la direzione della normale alla superficie, allora il campo vettoriale $\vec{t}$ delle forze di contatto locale si sviluppa lungo la stessa direzione. I $\vec{t}(\vec{x}, \hat{n})$ risultano essere tutti paralleli a $\hat{n}$ perciò, essendo $\vec{t} = \sigma\hat{n}$, σ deve essere un campo vettoriale di tensori scalari cioè di ordine zero. Conseguentemente è di facile intuizione che questo, per i fluidi non viscosi, sia l'opposto della pressione

$$\sigma = -p\delta \tag{4.3}$$

dove δ è il tensore identità e *p* la pressione.

Sostituendo la (4.1) nella (4.1) si ha

$$\vec{F} = \iint_{\partial\Omega} \sigma\hat{n}.$$

Per poter trasformare quest'ultima in un integrale di volume si può usare il noto teorema della divergenza, da cui

$$\vec{F} = \iiint_{\Omega} \nabla \cdot \sigma.$$

Ignorando la forza peso, sia *M* la massa del fluido e $\vec{A}$ l'accelerazione del suo centro di massa, allora

$$\vec{F} = M\vec{A} = \iiint_{\Omega} \rho \frac{D\vec{u}}{Dt}.$$

Eguagliando ambo i membri delle due equazioni precedenti si ha

$$\iiint_{\Omega} \rho \frac{D\vec{u}}{Dt} = \iiint_{\Omega} \nabla \cdot \sigma$$

e, valendo quest'ultima per qualsiasi volume scelto, allora gli integrandi devo essere uguali, cioè

$$\rho \frac{D\vec{u}}{Dt} = \nabla \cdot \sigma.$$

Aggiungendo a quest'ultima la forza peso esclusa poc'anzi si ottiene

$$\frac{D\vec{u}}{Dt} = \frac{1}{\rho}\vec{g} + \frac{1}{\rho}\nabla \cdot \sigma. \tag{4.4}$$

Per i fluidi non viscosi, sapendo che vale la (4.3) e ricordando la definizione di derivata materiale, tramite una banale sostituzione è possibile ottenere l'equazione del moto (1.1). Per i fluidi in generale, si utilizza come tensore σ la quantità

$$\sigma = -p\delta + \tau, \tag{4.5}$$

dove τ è un altro tensore simmetrico che permette di modellare la viscosità, mentre la pressione garantisce la condizione di incomprimibilità. A questo punto bisogna trovare un valore per il tensore appena introdotto e per farlo serve sapere come calcolare lo sforzo viscoso.

4.1.2 Sforzo viscoso

Lo sforzo della viscosità può essere definito a partire dalla componente della deformazione indotta dal flusso che, insieme alle informazioni sulla rotazione rigida, costituisce $\nabla\vec{u}$, cioè la variazione delle velocità locali. Tale componente di interesse, detta anche *velocità di deformazione*, si può ricavare dal prodotto scalare di due vettori del campo. Questo è possibile perché il campo stesso rappresenta la stima della variazione di velocità con cui il fluido si deforma. Si può dimostrare che la velocità di variazione del prodotto scalare di due vettori nel flusso, è data dalla parte simmetrica del gradiente della velocità. Si definisca *D* la matrice corrispondente a quest'ultima, allora

$$D = \frac{1}{2}(\nabla\vec{u} + \nabla\vec{u}^T).$$

D è chiamato *tensore velocità di deformazione* o semplicemente *tasso di deformazione* e, per i fluidi incomprimibili, la somma delle componenti della sua diagonale risulta essere

$$\nabla \cdot \vec{u} = 0.$$

Per trovare il tensore simmetrico τ introdotto poc'anzi, si può assumere che questo sia proporzionale al tasso di deformazione D. Questa proporzionalità, come già accennato all'inizio della trattazione, permette di distinguere tra fluidi newtoniani e fluidi non newtoniani. La relazione che lega τ a D per i fluidi newtoniani incomprimibili è la seguente

$$\tau = 2\mu D + \lambda tr(D)\delta, \tag{4.6}$$

dove μ è il coefficiente della viscosità dinamica e λ il secondo coefficiente della viscosità. Per i fluidi comprimibili $\lambda = -\frac{2}{3}\mu$, mentre per i fluidi incomprimibili, analizzati in questa trattazione, λ deve essere maggiore o uguale a $-\frac{2}{3}\mu$. Omettere tale vincolo, comporterebbe la violazione delle leggi della termodinamica di base conseguentemente, essendo il secondo termine della (4.6) uguale a zero per i fluidi incomprimibili e per qualsiasi valore di λ, allora è possibile scegliere $\lambda = 0$.

Infine, per i fluidi newtoniani, se nella simulazione viene trattata la temperatura, è possibile assumere μ come funzione di quest'ultima [CMIT02]. Si può notare poi come, aggiungendo la (4.6) all'equazione del moto, si ottenga

$$\frac{D\vec{u}}{Dt} + \frac{1}{\rho}\nabla p = \frac{1}{\rho}\nabla \cdot (\mu(\nabla\vec{u} + \nabla\vec{u}^T)). \tag{4.7}$$

Questa, ponendo μ costante e svolgendo qualche semplificazione permette di ottenere la (1.1). Tuttavia, se la simulazione prevede l'utilizzo di variabili relative alla viscosità, è necessario utilizzare esclusivamente la (4.7) in quanto, l'uso della (1.1), porterebbe ad una scorretta applicazione delle condizioni ai bordi.

Come per ogni argomento trattato, è necessario studiare come la viscosità appena introdotta, modifichi le condizioni ai bordi esposte nei capitoli precedenti. Dalla (4.5), risulta che la condizione ai bordi per superfici libere è

$$\sigma\hat{n} = -p\hat{n} + \tau\hat{n} = 0. \tag{4.8}$$

Da questa, se lo sforzo viscoso è nullo, cioè si sta simulando un fluido non viscoso, si ottiene $p = 0$. Nel caso in cui τ fosse diverso da zero, Batty e Bridson [BB08] hanno dimostrato che porre le due condizioni separate $p = 0$ e $\nabla\vec{u} \cdot \hat{n} = 0$ per la simulazione di fluidi altamente viscosi, eliminerebbe molti effetti visivi della simulazione stessa. Conseguentemente è bene utilizzare la (4.8) che, esplicitando le velocità, diventa

$$\sigma\hat{n} = -p\hat{n} + (\nabla\vec{u} + \nabla\vec{u}^T)\hat{n} = 0.$$

La condizione ai bordi per superfici solide, nel caso di fluidi altamente viscosi, è la già accennata condizione no-slip (1.20):

$$\vec{u} = \vec{u}^{solido}.$$

Questa, a differenza della condizione no-stick, calcola la velocità anche in direzione tangenziale rispetto alla superficie e non solo normale. E' quindi largamente utilizzata, in quanto verificata sperimentalmente come stabile e accurata rispetto alla condizione no-stick. Si fa presente, per completezza, che introduce la possibilità di aggiungere alla simulazione un effetto definito *strato limite* che però, graficamente, può essere trascurato. In questo strato, molto sottile e posizionato vicino alla superficie del solido, la velocità tangenziale varia repentinamente da un lato all'altro, passando dalla velocità $\vec{u}^{solido}$ a $\vec{u}^{*}$, ovvero la velocità che si sarebbe ottenuta ponendo le condizioni no-stick ai bordi per i fluidi non viscosi. Ricordando come vengono rappresentate graficamente le due condizioni sopra riportate, è infine possibile pensare all'effetto che lo strato limite introduce nella simulazione. Questo fa sì che l'effetto di scorrimento del fluido sul solido avvenga solo in una piccola regione vicino a quest'ultimo.

4.1.3 Implementazione

Come per la risoluzione delle equazioni di avvezione e delle forze del corpo, è conveniente separare i calcoli relativi alla viscosità dal resto dei metodi necessari per la simulazione dei fluidi. Il calcolo dell'effetto della viscosità sulle velocità va eseguito dopo aver reso incomprimibile il flusso, cioè dopo aver calcolato il gradiente della pressione. Separando questi due calcoli in metodi differenti, anche le condizioni ai bordi saranno applicate separatamente, cioè durante il calcolo della pressione si imporranno le (1.19) e (1.21), mentre per il calcolo della viscosità le (1.20) e (4.8) esplicitando le velocità. Ad ogni iterazione, nell'intervallo di tempo Δt, l'aggiornamento delle velocità, a seguito dell'applicazione della viscosità al campo, è dato dalla (4.7). Come per la pressione, dato il tensore dello sforzo viscoso τ, si può valutare il contributo dato da questo ad una componente della velocità, ad esempio quella orizzontale, come

$$u^{n+1} = u^{P} + \frac{\Delta t}{\rho}\left(\frac{\partial \tau^{11}}{\partial x} + \frac{\partial \tau^{12}}{\partial y} + \frac{\partial \tau^{13}}{\partial z}\right), \tag{4.9}$$

dove u^{P} indica il campo di velocità ottenuto a seguito del calcolo della pressione.

Conseguentemente, se *u* si trova nella griglia in posizione (i+1/2,j,k), ad esempio, allora si può chiedere che τ^{11} sia in posizione (i,j,k), τ^{12} in posizione (i+1/2,j+1/2,k) e τ^{13} in

posizione (i+1/2,j,k+1/2) da cui, sempre per la medesima componente della velocità, si ottiene dalla (4.9) la discretizzazione

$$\mathrm{u}_{i+\frac{1}{2},j,k}^{n+1} = \mathrm{u}_{i+\frac{1}{2},j,k}^{P} + \frac{\Delta t}{\rho}\left(\frac{\tau_{i+1,j,k}^{11} - \tau_{i,j,k}^{11}}{\Delta x} + \frac{\tau_{i+\frac{1}{2},j+\frac{1}{2},k}^{12} - \tau_{i+\frac{1}{2},j-\frac{1}{2},k}^{12}}{\Delta y} + \frac{\tau_{i+\frac{1}{2},j,k+\frac{1}{2}}^{13} - \tau_{i+\frac{1}{2},j,k-\frac{1}{2}}^{13}}{\Delta z}\right).$$

E' necessario capire, a questo punto, come ottenere i valori delle componenti del tensore dello sforzo viscoso dalla griglia. Un primo metodo è quello delle differenze centrali sul campo di velocità dato. Tale approccio introduce svariati problemi perché necessita dell'utilizzo di valori fantasma, già trattati nel corso di questo documento, per le celle di aria interessate dal calcolo. Questo porta all'uso di tecniche, quali l'estrapolazione lineare, che risultano molto instabili per valori dell'intervallo di tempo Δt grandi. Per evitare che una certa quantità q, a causa di questa instabilità, assuma un comportamento oscillatorio o di crescita esponenziale bisogna porre il limite

$$\Delta \mathrm{t} = \frac{\Delta x^2 \rho}{12\mu_{max}},$$

cioè una restrizione pari a un ordine di grandezza superiore a $O(\Delta x)$ necessario per il controllo degli errori nell'avvezione. Una soluzione numerica, spesso utilizzata per risolvere il problema sopra introdotto, è quella *dell'integrazione del tempo implicita* tramite il metodo di Eulero all'indietro. Si valuta il tensore dello sforzo viscoso sulla base dei nuovi valori delle velocità $\vec{u}^{n+1}$ che si calcolano a partire dal tensore stesso. Questa definizione implicita delle nuove velocità permette di utilizzare la discretizzazione di Eulero all'indietro, cioè

$$q_i^{n+1} = q_i^n + \Delta t k \frac{q_{i+1}^{n+1} - 2q_i^{n+1} + q_{i-1}^{n+1}}{\Delta x^2}$$

che, insieme alle condizioni no-slip, produce un sistema lineare con matrice simmetrica definita positiva risolvibile con PCG. Questo approccio implicito assicura l'assenza di crescite incontrollate di valori e la stabilità incondizionata qualsiasi sia il valore Δt. Benché l'aggiunta della condizione ai bordi per le superfici libere $(\nabla\vec{u} + \nabla\vec{u}^T)\hat{n} = 0$ all'interno di tale metodo risulta molto complicata Batty e Bridson [BB08], utilizzando lo strumento matematico del *calcolo delle variazioni,* hanno riformulato il metodo implicito di Eulero all'indietro per poter calcolare un campo di velocità minimizzante una data quantità. Un vantaggio di tale metodo è quello di comprendere già al suo interno le condizioni ai bordi per superfici libere. Si tralascia in questa trattazione la parte puramente matematica, che è possibile consultare nel testo di Batty e Bridson, per esporre la parte implementativa. La quantità che si vuole minimizzare è l'energia così definita

$$E[\vec{u}] = \iiint_\Omega \frac{\rho}{2}\|\vec{u} - \vec{u}^P\|^2 + \iiint_\Omega \Delta t\mu \left\|\frac{\Delta\vec{u} - \Delta\vec{u}^P}{2}\right\|_F^2 .$$

Il nuovo campo di velocità tende ad essere in equilibrio tra due comportamenti, l'esser vicino a $\vec{u}^P$ e il voler eliminare tutte le deformazioni in modo da ottenere un movimento rigido. Tale equilibrio è definito dalla densità, l'intervallo di tempo e il coefficiente di viscosità. Per trovare l'energia $E[\vec{u}]$ è necessario approssimarla come somme e differenze finite, considerando tutti i punti della griglia sfalsata che sono all'interno del fluido.

Di seguito si mostra lo svolgimento del primo integrale, mentre del secondo la scomposizione in componenti distinte della *norma di Frobenius*. Di quest'ultima scomposizione va eseguita poi la stessa operazione effettuata sul primo integrale.

$$\iiint_\Omega \frac{\rho}{2}\|\vec{u} - \vec{u}^P\|^2 = \iiint_\Omega \frac{\rho}{2}(u - u^P)^2 + \iiint_\Omega \frac{\rho}{2}(v - v^P)^2 + \iiint_\Omega \frac{\rho}{2}(w - w^P)^2$$

$$\approx \sum_{\left(i+\frac{1}{2},j,k\right)\in\Omega} \frac{\rho_{i+\frac{1}{2},j,k}}{2}\left(u_{i+\frac{1}{2},j,k} - u^P_{i+\frac{1}{2},j,k}\right)^2 \Delta x^3$$

$$+ \sum_{\left(i,j+\frac{1}{2},k\right)\in\Omega} \frac{\rho_{i,j+\frac{1}{2},k}}{2}\left(v_{i,j+\frac{1}{2},k} - v^P_{i,j+\frac{1}{2},k}\right)^2 \Delta x^3$$

$$+ \sum_{\left(i,j,k+\frac{1}{2}\right)\in\Omega} \frac{\rho_{i,j,k+\frac{1}{2}}}{2}\left(w_{i,j,k+\frac{1}{2}} - w^P_{i,j,k+\frac{1}{2}}\right)^2 \Delta x^3$$

$$\left\|\frac{\Delta\vec{u} - \Delta\vec{u}^P}{2}\right\|_F^2 = u_x^2 + \frac{1}{4}(u_y + v_x)^2 + \frac{1}{4}(u_z + w_x)^2 + v_y^2 + \frac{1}{4}(v_z + w_y)^2 + w_z^2$$

Una volta calcolate le somme, è possibile differenziarle rispetto le velocità sconosciute e ponendo il gradiente uguale a zero, trovando il "*minimizer"* E.

Il sistema ottenuto è formato da equazioni lineari con matrice definita semi-positiva, conseguentemente è possibile applicare il metodo PCG per risolverlo. I valori di ρ sono memorizzati all'interno della griglia e, nel caso questi non siano sui punti di quest'ultima, è necessario fare una media per ottenere il loro valore in punti sfasati. I valori delle velocità che risiedono nel solido vanno sostituite con le velocità dello stesso, mentre per le velocità che risiedono in celle di aria andranno eliminati.

L'uso di griglie sfalsate, come per tutti i metodi presentati nel capitolo 2, rende più efficiente il calcolo della viscosità.

Infine, è doveroso far presente che il metodo di cui si forniscono le tecniche implementative in questo paragrafo, per il calcolo della viscosità, può essere unito al metodo trattato nel paragrafo 2.3 relativo al calcolo della pressione. Batty e Bridson [BB10] hanno introdotto un metodo ibrido, infatti, che garantisce l'incomprimibilità del fluido e, contemporaneamente, integra gli effetti viscosi dello stesso. Tale metodo è spesso definito come *flusso di Stokes instabile* e in questa trattazione non viene proposto, in quanto si preferisce separare le varie parti di calcolo dell'algoritmo di simulazione dei fluidi.

4.2 Turbolenza

Un altro effetto, soprattutto nella simulazione di oceani, che può essere utile trattare è la turbolenza, cioè quell'effetto prodotto dalla presenza di vortici all'interno del fluido. Per arricchire il metodo di Navier-Stokes in modo tale che possa rappresentare anche questo aspetto grafico, è necessario introdurre il concetto di *vorticità* relativo ad un campo di velocità.

4.2.1 L'equazione della vorticità ed il suo confinamento

Come accennato nel paragrafo precedente, il gradiente di un campo di velocità fornisce una matrice la cui parte simmetrica misura la sua deformazione, mentre la parte antisimmetrica la sua rotazione. Più precisamente, nella parte antisimmetrica della matrice sopra citata, sono presenti le componenti della velocità angolare del campo. Sia definito

$$\vec{u}(\vec{x}) = \vec{U} + \vec{\Omega} \times \vec{x} \tag{4.10}$$

il campo di velocità relativo ad un moto rigido, nel quale $\vec{U}$ è la traslazione e $\vec{\Omega}$ la velocità angolare attorno all'origine. Il gradiente del campo definito dalla (4.10) risulta essere

$$\frac{\partial \vec{u}}{\partial \vec{x}} = \frac{\partial}{\partial \vec{x}} \begin{pmatrix} U_1 + \Omega_2 z - \Omega_3 y \\ U_2 + \Omega_3 x - \Omega_1 z \\ U_3 + \Omega_1 y - \Omega_2 x \end{pmatrix} = \begin{pmatrix} 0 & -\Omega_3 & \Omega_2 \\ \Omega_3 & 0 & -\Omega_1 \\ -\Omega_2 & \Omega_1 & 0 \end{pmatrix}.$$

Da questa, considerando la parte antisimmetrica $\frac{1}{2}\left(\frac{\partial \vec{u}}{\partial \vec{x}} - \frac{\partial \vec{u}^T}{\partial \vec{x}}\right)$, è possibile ottenere un valore per la velocità angolare in un moto rigido, come

$$\vec{\Omega}(\vec{x}) = \frac{1}{2}\left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}, \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}, \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right),$$

che è pari alla metà della vorticità. Quindi si definisce la vorticità $\vec{\omega}$ come il *rotore* di un campo vettoriale, cioè

$$\vec{\omega} = \nabla \times \vec{u} = \left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}, \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}, \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right).$$

E' necessario indicare che, la vorticità in un fluido incomprimibile, avviene solo in presenza di contorni solidi e dalla *condizione di aderenza* no-stick introdotta precedentemente. Specificato ciò, grazie ad una serie di modifiche, è possibile trasformare la (1.1), assumendo viscosità costante, *nell'equazione della vorticità*

$$\frac{\partial \vec{\omega}}{\partial t} + \vec{u} \cdot \nabla \vec{\omega} = -\omega \cdot \nabla \vec{u} + \nu \nabla \cdot \nabla \vec{\omega}, \tag{4.11}$$

dove $\omega \cdot \nabla \vec{u}$, definito *"vortex-stretching",* è un termine geometrico pari a

$$\omega \cdot \nabla \vec{u} = \begin{pmatrix} \omega_1 \frac{\partial u}{\partial x} + \omega_2 \frac{\partial v}{\partial x} + \omega_3 \frac{\partial w}{\partial x} \\ \omega_1 \frac{\partial u}{\partial v} + \omega_2 \frac{\partial v}{\partial v} + \omega_3 \frac{\partial w}{\partial v} \\ \omega_1 \frac{\partial u}{\partial z} + \omega_2 \frac{\partial v}{\partial z} + \omega_3 \frac{\partial w}{\partial z} \end{pmatrix}.$$

Il calcolo della vorticità non rimane esente dalla diffusione numerica. Quella introdotta dall'avvezione euleriana, può essere superata mediante l'utilizzo del metodo lagrangiano FLIP. Tuttavia, a questa fonte di diffusione numerica, si unisce quella causata dall'utilizzo di algoritmi separati per il calcolo dell'avvezione, della pressione e dell'aggiornamento delle velocità. Questa seconda diffusione, provoca la riduzione, a livello grafico, della vorticità del campo. E' possibile verificare, infatti, come a partire da una rotazione rigida avente vorticità ω, un'iterazione dell'avvezione la cambierebbe in $\omega \cos\left(\frac{\omega \Delta t}{2}\right)$. Vanno adottate, quindi, tecniche che mantengano la vorticità inalterata durante le iterazioni dell'algoritmo. La tecnica di *confinamento della vorticità*, studiata da Steinhoff e Underhill [SU94] e utilizzata per la prima volta da Fedkiw [FSJ01] è una modifica delle equazioni di Navier-Stokes che permette di fare ciò. L'idea alla base del contenimento è quella di individuare i vortici, all'interno del fluido, ed applicarvi forze che mantengano inalterato il loro moto rotazionale. Siano $\vec{N}$ dei vettori unitari, direzionati verso il centro rotazionale di un vortice, ottenuti come segue

$$\vec{N} = \frac{\nabla \|\vec{\omega}\|}{\|\nabla \|\vec{\omega}\|\|}, \tag{4.12}$$

sapendo che $\vec{\omega}$ punta lungo l'asse di rotazione, allora è possibile calcolare la forza sopra citata come

$$f_{conf} = \epsilon \Delta x (\vec{N} \times \vec{\omega}), \tag{4.12}$$

dove ϵ è un parametro di controllo che permette di aggiustare il confinamento.

Siccome Δx, per miglioramenti successivi della risoluzione della griglia, tende a zero, allora anche la dissipazione dell'errore numerico della vorticità tende a zero. Per risolvere

la (4.12) si procede come si è visto nel corso di tutta la trattazione: dalla griglia MAC si ottengono le velocità, si usa il metodo delle differenze centrali per approssimare la vorticità e per stimare il gradiente, e infine la si risolve. Di seguito sono mostrati i passaggi.

$$\vec{\omega}_{i,j,k} = \left(\frac{w_{i,j+1,k} - w_{i,j-1,k}}{2\Delta x} - \frac{v_{i,j+1,k} - v_{i,j-1,k}}{2\Delta x}, \frac{u_{i,j,k+1} - u_{i,j,k-1}}{2\Delta x} - \frac{w_{i+1,j,k} - w_{i-1,j,k}}{2\Delta x}, \frac{v_{i+1,j,k} - v_{i-1,j,k}}{2\Delta x} - \frac{u_{i,j+1,k} - u_{i,j-1,k}}{2\Delta x}\right)$$

$$\nabla\|\vec{\omega}\|_{i,j,k} = \left(\frac{\|\vec{\omega}\|_{i+1,j,k} - \|\vec{\omega}\|_{i-1,j,k}}{2\Delta x}, \frac{\|\vec{\omega}\|_{i,j+1,k} - \|\vec{\omega}\|_{i,j-1,k}}{2\Delta x}, \frac{\|\vec{\omega}\|_{i,j,k+1} - \|\vec{\omega}\|_{i,j,k-1}}{2\Delta x}\right)$$

$$\vec{N} = \frac{\nabla\|\vec{\omega}\|_{i,j,k}}{\left\|\nabla\|\vec{\omega}\|_{i,j,k}\right\| + 10^{-20}M}$$

Nella formula sopra riportata, *M* è un valore introdotto per assicurarsi che il termine a denominatore sia diverso da zero e, solitamente, corrisponde a $1/(\Delta x \Delta t)$.
La tecnica del confinamento della vorticità non permette di modificare selettivamente la vorticità di un vortice, così come non ne permette l'introduzione di nuova. Per svolgere questi compiti si può utilizzare la *turbolenza procedurale*.

4.2.2 Turbolenza procedurale

L'idea alla base delle tecniche di turbolenza procedurale è quella di creare, proceduralmente, un campo di velocità di turbolenza che permetta di introdurre, su una scala minore rispetto alla simulazione, falsi dettagli aggiuntivi senza aumentare ulteriormente il costo della simulazione stessa.
Per l'uso di una griglia *n*x*n*x*n*, infatti, sarà necessaria $O(n^3)$ memoria, alla quale va aggiunta tutta quella richiesta per l'allocazione dei vettori utilizzati nei vari step dell'algoritmo. Mantenendo Δt proporzionale a Δx e usando MICCG(0), che richiede $O(n^{1/2})$ iterazioni per convergere, si ottiene un costo totale di $O(n^{4.5})$. E' quindi necessario evitare ulteriori costi aggiuntivi.

Il campo generato come sopra deve garantire sia il controllo sulla velocità rispetto alle diverse scale di lunghezza, sia che le velocità rimangano a divergenza nulla. Un primo metodo per la creazione di campi di velocità turbolenti si basa sull'uso dello *spazio di*

Fourier. Sia $\vec{u}(\vec{x})$ un campo di velocità periodico su un cubo di lato *L*, questo, facendone la trasformata di Fourier, può essere riscritto nel seguente modo

$$\vec{u}(\vec{x}) = \sum_{i,j,k=-\infty}^{\infty} \hat{u}_{i,j,k} e^{\sqrt{-1}2\pi(ix+jy+kz)/L}.$$

dove i coefficienti di Fourier $\hat{u}_{i,j,k}$ sono vettori tridimensionali nel campo complesso. Ciò non introduce notevole complessità, in quanto su questi verrà utilizzata la trasformata di Fourier. Il modello *Kolmogorov "5/3-law"*, introdotto da Shinya e Fournier [SF92], è un modello molto semplice. Questo afferma che, in presenza di una turbolenza in uno stato stazionario, l'energia cinetica contenuta in tutte le modalità di Fourier delle frequenze spaziali attorno *ω* dovrebbero scalare secondo $\omega^{-\frac{5}{3}}$, cioè che il coefficiente di Fourier (*i*,*j*,*k*) deve avere una grandezza dell'ordine di

$$\left\|\hat{u}_{i,j,k}\right\| \sim (i^2 + j^2 + k^2)^{-11/12}.$$

La divergenza della velocità è data da

$$\nabla\vec{u}(\vec{x}) = \sum_{i,j,k=-\infty}^{\infty} \frac{\sqrt{-1}2\pi}{L}[(i,j,k) \cdot \hat{u}_{i,j,k}]e^{\sqrt{-1}2\pi(ix+jy+kz)/L}.$$

Una volta ottenuti i coefficienti di Fourier, è possibile applicare la *trasformata di Fourier inversa veloce* (FFT) su ogni componente per ottenere una griglia di velocità che però non risulterà sfalsata. Una volta fatta l'avvezione, per animare il tutto si possono costruire i due campi di velocità ed attuare la *dissolvenza incrociata avanti e indietro*, introdotta da Rasmussen [RNFG03]. L'animazione così ottenuta pecca di plausibilità che, tuttavia è "mascherata" dall'aggiunta di questa ad una simulazione dettagliata già esistente. Il metodo sopra definito, chiamato a volte *sintesi di Fourier*, possiede diversi pregi, però non permette contemporaneamente il controllo locale dell'intensità dei vortici e la garanzia della divergenza nulla del campo costruito.

Un metodo alternativo che assicura la divergenza nulla, generando il campo tramite la creazione di blocchi come, ad esempio, il *Perlin noise* richiede che la condizione sulla divergenza si ottenga dalle identità del calcolo vettoriale.

Kniss e Hart [KH04] utilizzano la seguente identità per la quale, per ogni campo vettoriale $\vec{\psi}$

$$\nabla \cdot \left(\nabla \times \vec{\psi}\right) = 0$$

da cui, considerando $\vec{\psi}$ una funzione di rumore valutata su vettori, si ottiene

$$\vec{\psi}(\vec{x}) = \sum_{p=1}^{m} A_p \vec{N}\left(\frac{C2^p\vec{x}}{\Delta x}\right)$$

dove A_p è l'ampiezza dell'ottava di rumore, che opportunamente modificata permette di agire localmente sull'intensità del vortice specifico.
Il campo di velocità viene quindi calcolato come

$$\vec{u} = \nabla \times \vec{\psi},$$

nel quale il rotore può essere approssimato con le differenze finite.
Negli ultimi anni, grazie ai numerosi studi effettuati su questo campo, sono state introdotte altre tecniche che vanno a colmare i difetti insiti nei metodi sopra trattati, grazie all'utilizzo di sotto griglie per la simulazione della turbolenza. Data la scarsità di utilizzi pratici e la continua ricerca che si sta svolgendo su quest'ultimi, però, si preferisce evitarne la trattazione, rimandando a ricerche future lo studio degli stessi.

4.3 Acque basse

Con i paragrafi precedenti, si è conclusa l'esposizione dei metodi che permettono effetti più avanzati per la simulazione di fluidi. In questo e nel prossimo, si vuole trattare la rappresentazione di due casi particolari relativi all'acqua. In questo paragrafo si mostra, quindi, l'impostazione e la tecnica implementativa per la simulazione di acque basse. Con questo termine si intendono quegli ambienti nei quali l'acqua si trova o arriva ad avere una profondità limitata. Tale condizione implica delle semplificazioni, sia nella modellazione che nella conseguente tecnica implementativa.

4.3.1 Introduzione al metodo e premesse

Una prima premessa, che sarà utile anche per il paragrafo successivo sugli oceani, è che la superficie dell'acqua o interfaccia, sia definita da un *campo altezza* $y = h(x, z)$. Conseguentemente, se *y* è minore di tale quantità, allora la regione comprenderà l'acqua, altrimenti l'aria. Analogamente all'interfaccia, anche il "fondale" può essere definito da un campo altezza come $y = b(x, z)$ e di conseguenza è possibile definire il volume di acqua come

$$b(x, z) < y < h(x, z).$$

E' di banale intuizione che anche la profondità può, a questo punto, essere vista come un campo altezza nel seguente modo

$$d(x, z) = h(x, z) - b(x, z).$$

La definizione della profondità come sopra permette di ricostruire facilmente l'interfaccia in funzione degli altri due campi. E' possibile poi assumere che il fondale non si muova,

cioè che $b(x,z)$ rimanga costante. Per fare in modo poi che il campo altezza descriva al meglio la profondità dell'acqua per tutta la simulazione, si deve considerare il fatto che le velocità non siano troppo elevate. Un'altra assunzione è data dalla definizione stessa di acqua bassa, cioè che la profondità *d* non sia molto alta. Ciò comporta logicamente la possibilità di ignorare le variazioni verticali nel campo delle velocità, permettendo di lavorare con le loro componenti orizzontali ad una profondità media, cioè $u(x,z)$ e $w(x,z)$, sono le medie di u e w rispettivamente al variare di y. I sistemi di equazioni, risultanti dalla media delle velocità rispetto alla profondità, vengono definiti *equazioni delle acque basse*. Supponendo, infine, che l'acqua sia calma e che sia presente la pressione idrostatica è possibile dall'equazione del moto seguente

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} + \frac{1}{\rho}\frac{\partial p}{\partial y} = -g,$$

ottenere

$$\frac{1}{\rho}\frac{\partial p}{\partial y} = -g,$$

in quanto la componente della pressione influenza maggiormente il moto dell'acqua.
Da cui, con l'aggiunta della condizione ai bordi $p = 0$ e ponendo $y = h$ si ricava

$$p(x,y,z,t) = \rho g(h(x,z,t) - y). \tag{4.13}$$

Questa uguaglianza semplifica il costo computazionale, in quanto evita di risolvere il sistema lineare introdotto nel secondo capitolo, essendo un calcolo costoso.
Grazie alle considerazioni fatte poc'anzi, le componenti orizzontali della velocità si riducono in modo banale. Annullandosi le derivate parziali rispetto alla *y*, infatti, l'equazione del momento perde la corrispondente componente del gradiente, trasformando l'avvezione tridimensionale fino ad ora trattata in avvezione bidimensionale. Se in queste viene poi sostituita la (4.13), allora

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + w\frac{\partial u}{\partial z} + g\frac{\partial h}{\partial x} = 0,$$
$$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + w\frac{\partial w}{\partial z} + g\frac{\partial h}{\partial z} = 0. \tag{4.14}$$

Ciò però non cambia la tecnica avvettiva che rimane la stessa usata in precedenza. Grazie ad una serie di considerazioni, è possibile affermare che sul fondale, cioè $y = b(x,z)$, essendo la componente normale a quest'ultimo proporzionale a $(-\partial b/\partial x\,, 1, -\partial b/\partial z)$ allora

$$v = u\frac{\partial b}{\partial x} + w\frac{\partial b}{\partial z}.$$

Da questa è facile intuire che, se il fondale è piano, allora *v* è pari a zero, mentre per qualsiasi altro punto, sostituendo il risultato sopra alla condizione di incomprimibilità si ottiene che

$$v(x,y,z,t) = u\frac{\partial b}{\partial x} + w\frac{\partial b}{\partial z} - \left(\frac{\partial u}{\partial x} + \frac{\partial u}{\partial z}\right)(y-b). \tag{4.15}$$

La (4.15) permette di scegliere *v* in modo da garantire la condizione di incomprimibilità e le condizioni ai bordi per superfici solide sul fondale.

La funzione $\phi(x,y,z) = y - h(x,z)$ definisce l'interfaccia dell'acqua che dovrà muoversi con la velocità del fluido. Ciò significa chiedere che la funzione ϕ soddisfi l'equazione di avvezione, perciò

$$-\frac{\partial h}{\partial t} + u\left(-\frac{\partial h}{\partial x}\right) + v(1) + w\left(-\frac{\partial h}{\partial z}\right) = 0,$$

per $y = h$. Sostituendo quest'ultima relazione nella (4.15) si ottiene l'equazione che descrive la variazione del cambio di altezza, cioè

$$\frac{\partial h}{\partial t} + u\frac{\partial (h-b)}{\partial x} + w\frac{\partial (h-b)}{\partial z} = -(h-b)\left(\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z}\right).$$

Ricordando che $d = h - b$ e che *b* è stazionario, allora l'equazione sopra si semplifica in

$$\frac{\partial d}{\partial t} + u\frac{\partial d}{\partial x} + w\frac{\partial d}{\partial z} = -d\left(\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z}\right), \tag{4.16}$$

in cui il primo membro corrisponde all'avvezione della profondità, la quale risulta proporzionale alla profondità stessa e alla divergenza bidimensionale. La (4.16) re-arrangiata corrisponde alla legge di conservazione del volume, cioè

$$\frac{\partial d}{\partial t} + \frac{\partial}{\partial x}(ud) + \frac{\partial}{\partial z}(wd) = 0$$

che permette di arrivare a metodi numerici che conservano, appunto, il volume totale dell'acqua nel sistema.

Infine, dopo aver studiato come modificare le velocità e la profondità o altezza dell'interfaccia dell'acqua, è necessario passare alle condizioni ai bordi. Le equazioni finora viste contemplano già le condizioni ai bordi per superfici libere come, ad esempio, l'interfaccia dell'acqua. Le uniche condizioni ai bordi non considerate riguardano i confini, sul piano orizzontale, dove l'acqua finisce e quelli dove finisce il dominio. Sia $\hat{n}$ la normale bidimensionale al confine solido sul piano orizzontale, allora la condizione ai bordi risulta

$$(u,w)\cdot\hat{n} = 0.$$

Nel caso il confine solido abbia una sua velocità, la condizione sopra diventa

$$(u_{solido}, w_{solido})\cdot\hat{n} = 0$$

da cui, per mantenere la velocità nella direzione normale è necessario che

$$\left(\frac{\partial h}{\partial x}, \frac{\partial h}{\partial z}\right) \cdot \hat{n} = 0.$$

La modellazione effettuata gestisce un'ultima condizione, sulla *linea di contatto*, dove la profondità cala a zero

4.3.2 L'equazione delle onde

Prima di passare al metodo numerico per la simulazione di acque basse, è possibile fare un'ultima considerazione e semplificazione. Per acque calme, come supposto all'inizio del paragrafo, l'avvezione può essere ignorata. Questo perché, per questo tipo di acque, le particelle tendono a non muoversi e da quello che si è detto, già nel primo capitolo, l'avvezione misura proprio come avviene questo movimento. Le equazioni corrispondenti alle componenti orizzontali della velocità (4.14) diventano

$$\frac{\partial u}{\partial t} + g\frac{\partial h}{\partial x} = 0,$$
$$\frac{\partial w}{\partial t} + g\frac{\partial h}{\partial z} = 0,$$

mentre la (4.16) diventa

$$\frac{\partial h}{\partial t} = -d\left(\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z}\right).$$

Da questa, si ottiene dopo banali operazioni

$$\frac{\partial^2 h}{\partial t} = g\mathrm{d}\nabla \cdot \nabla\mathrm{h}$$

che, sotto l'ipotesi che la profondità rimanga costante, viene chiamata *equazione delle onde* e le sue soluzioni indicano la velocità con le quali le onde si muovono.
Il modello così definito risulta pienamente in linea con il comportamento delle onde la cui velocità, ad esempio, aumenta con l'aumentare della profondità.

Per concludere la trattazione sulle acque basse, si vuole mostrare l'algoritmo la loro simulazione. Una possibile discretizzazione è stata introdotta nel 2002 da Layton e Van de Panne [LVDP02] e risulta incondizionatamente stabile. Tuttavia, introducendo un semplice vincolo all'intervallo di tempo, è possibile evitare la soluzione del sistema lineare, risparmiando sul costo generale dell'algoritmo. Come per tutti i metodi esposti in questa trattazione, si utilizza una griglia MAC sfalsata nella quale vengono memorizzate le componenti della velocità sulle facce delle celle e la profondità al centro di quest'ultime. Supponiamo, come accennato all'inizio del paragrafo, che l'altezza sia

trovata a partire dalla profondità e dal fondale. Infine, per assicurare la stabilità, come accennato poc'anzi si ponga il vincolo

$$\Delta t < \frac{\Delta x}{\sqrt{gD}},$$

con *D* il massimo valore della profondità nella simulazione.

L'algoritmo è il seguente:

1. Si svolge l'avvezione con uno dei metodi trattati precedentemente, semi-lagrangiano o particellare.
2. Si calcola il campo di altezze intermedio $h^A = b + d^A$ e si estrapola alle celle non di fluido.
3. Si aggiornano le velocità con l'accelerazione della pressione

$$u^{n+1}_{i+\frac{1}{2},k} = u^{A}_{i+\frac{1}{2},k} - \Delta t g \frac{h^A_{i+1,k} - h^A_{i,k}}{\Delta x}$$

$$w^{n+1}_{i,k+\frac{1}{2}} = w^{A}_{i,k+\frac{1}{2}} - \Delta t g \frac{h^A_{i,k+1} - h^A_{i,k}}{\Delta x}$$

4. Si estrapolano queste nuove velocità alle celle non di fluido e si aggiorna la profondità con la componente della divergenza

$$d^{n+1}_{i,k} = d^A_{i,k} - \Delta t d^A_{i,k} \left(\frac{u^{n+1}_{i+1/2,k} - u^{n+1}_{i-1/2,k}}{\Delta x} + \frac{w^{n+1}_{i,k+1/2} - w^{n+1}_{i,k-1/2}}{\Delta x} \right)$$

4.4 Oceani

In questo paragrafo si vuole mostrare, utilizzando molti dei concetti trattati nei precedenti capitoli, come sia possibile effettuare una simulazione al calcolatore di un oceano. Di seguito, verranno effettuate molte semplificazioni, che permetteranno di ottenere un metodo veloce e poco costoso per svolgere tale simulazione. I metodi proposti in questo paragrafo, a seguito delle semplificazioni sopra citate, riguarderanno oceani relativamente calmi [TES04]. La difficoltà maggiore sarà data, quindi, dalla scelta del parametro della scala, in quanto lo studio delle onde deve concentrarsi sui fenomeni di larga scala e non. Altro elemento fondamentale nello studio di un metodo per la simulazione di un oceano, è la profondità dell'acqua, che permette di calcolare le velocità relative alle onde aventi differenti dimensioni, scale.

4.4.1 Flusso potenziale

Si consideri un oceano come sopra che, essendo calmo, possiede una velocità pari a zero e conseguentemente una vorticità nulla. Richiedere quindi che l'oceano sia calmo, equivale a chiedere che sia *irrotazionale*, cioè

$$\nabla \times \vec{u} = \vec{\omega} = 0.$$

Un teorema del calcolo vettoriale afferma che un campo vettoriale che possiede un rotore nullo in una *regione semplicemente connessa* corrisponde al gradiente di un potenziale scalare

$$\vec{u} = \nabla\phi, \tag{4.17}$$

dove ϕ è il suddetto potenziale scalare, ovvero un campo scalare.

Sostituendo il risultato ottenuto nella condizione di incomprimibilità si ottiene un'equazione Laplaciana che deve essere soddisfatta da ϕ, cioè

$$\nabla \cdot \nabla\phi = 0.$$

Questo risultato è molto importante in quanto è alla base del cosiddetto *flusso potenziale* e perché, supponendo un fluido irrotazionale e una regione semplicemente connessa, permette di ottenere con la risoluzione di una sola PDE la stessa soluzione che si sarebbe ottenuta risolvendo le equazioni non lineari di Navier-Stokes (1.1) e (1.2).

Come sempre è necessario studiare le condizioni ai bordi da applicare anche all'attuale simulazione. Per le condizioni ai bordi per superfici solide si ha la (1.19) che diventa un vincolo su $\nabla\phi \cdot \hat{n}$. Per quelle relative alle superfici libere, si deve compiere più lavoro. Sia

$$\frac{\partial \nabla\phi}{\partial t} + (\nabla\phi) \cdot (\nabla\nabla\phi) + \frac{1}{\rho}\nabla\phi = \vec{g}$$

l'equazione del moto per i fluidi non viscosi alla quale si è applicata la (4.17), vedendo l'accelerazione gravitazionale come il potenziale gravitazionale, cioè

$$\vec{g} \cdot \vec{x} = -gy$$

dove *y* indica l'altezza, allora

$$\nabla\frac{\partial\phi}{\partial t} + (\nabla\phi) \cdot (\nabla\nabla\phi) + \nabla\frac{p}{\rho} + \nabla(gy) = 0. \tag{4.18}$$

E' possibile notare come è presente il termine della pressione necessario per definire la condizione ai bordi per le superfici libere. Dalla (4.18), essendo

$$(\nabla\phi)\cdot(\nabla\nabla\phi)=\begin{pmatrix}\frac{\partial}{\partial x}\frac{1}{2}\left(\frac{\partial\phi}{\partial x}\right)^2+\frac{\partial}{\partial x}\frac{1}{2}\left(\frac{\partial\phi}{\partial y}\right)^2+\frac{\partial}{\partial x}\frac{1}{2}\left(\frac{\partial\phi}{\partial z}\right)^2\\ \frac{\partial}{\partial y}\frac{1}{2}\left(\frac{\partial\phi}{\partial x}\right)^2+\frac{\partial}{\partial y}\frac{1}{2}\left(\frac{\partial\phi}{\partial y}\right)^2+\frac{\partial}{\partial y}\frac{1}{2}\left(\frac{\partial\phi}{\partial z}\right)^2\\ \frac{\partial}{\partial z}\frac{1}{2}\left(\frac{\partial\phi}{\partial x}\right)^2+\frac{\partial}{\partial z}\frac{1}{2}\left(\frac{\partial\phi}{\partial y}\right)^2+\frac{\partial}{\partial z}\frac{1}{2}\left(\frac{\partial\phi}{\partial z}\right)^2\end{pmatrix}=\nabla\left(\frac{1}{2}\|\nabla\phi\|^2\right)$$

allora

$$\nabla\left[\frac{\partial\phi}{\partial t}+\frac{1}{2}\|\nabla\phi\|^2+\frac{p}{\rho}+\mathrm{gy}\right]=0.$$

Siccome il gradiente di una funzione è pari a zero solo se questa è una costante, allora si ottiene

$$\frac{\partial\phi}{\partial t}+\frac{1}{2}\|\nabla\phi\|^2+\frac{p}{\rho}+\mathrm{gy}=0$$

definita comunemente *equazione di Bernoulli*. In questo modo, come già specificato, è possibile collegare, tramite un'equazione non lineare, la pressione al potenziale ϕ. Dall'equazione di Bernoulli è possibile, sia trovare la pressione in un punto interno al fluido dato ϕ, sia la condizione ai bordi per superfici libere, ponendo $p=0$. Risolvere il flusso potenziale, ora che comprende una condizione ai bordi data da una equazione non lineare, necessita di ulteriori semplificazioni del modello iniziale.

4.4.2 Semplificazioni e risoluzione

Si supponga, quindi, che siano escluse dalla simulazione il fenomeno delle onde che si infrangono, in modo da definire l'interfaccia del mare come $y=h(x,z)$. Facendo le stesse considerazioni, su tale aspetto, svolte nel paragrafo precedente, l'avvezione si ottiene allo stesso modo che per le acque basse. Si suppone inoltre che il fondale sia piatto, cioè $y=-H$ con H molto grande. Facendo tali considerazioni è possibile riscrivere l'insieme delle equazioni differenziali da risolvere

$$\begin{aligned}\nabla\cdot\nabla\phi&=0 && \text{per } -H\le y\le h(x,z),\\ \tfrac{\nabla\phi}{\partial y}&=0 && \text{a } y=-H,\\ \tfrac{\partial\phi}{\partial t}+\tfrac{1}{2}\|\nabla\phi\|^2+\tfrac{p}{\rho}+gh(x,z)&=0 && \text{a } y=h(x,z),\\ \tfrac{\partial h}{\partial t}+(u,w)\cdot\left(\tfrac{\partial h}{\partial x},\tfrac{\partial h}{\partial z}\right)&=v.\end{aligned}$$

Da queste è possibile, grazie alle semplificazioni introdotte, eliminare termini che rispetto ad altri sono ininfluenti, quali $\vec{u}$ e h. Inoltre, considerando h molto piccolo, ponendo le condizioni ai bordi per $y=0$, allora si ottengono

$$\nabla \cdot \nabla\phi = 0 \qquad \text{per } -H \leq y \leq 0,$$

$$\frac{\nabla\phi}{\partial y} = 0 \qquad \text{a } y = -H,$$

$$\frac{\partial\phi}{\partial t} = -gh(x,z) \qquad \text{a } y = 0,$$

$$\frac{\partial h}{\partial t} = \frac{\nabla\phi}{\partial y} \qquad \text{a } y = 0.$$

Queste possono essere risolte come PDE, ottenendo quel miglioramento in performance accennato poc'anzi.

Le ultime condizioni ai bordi da considerare sono quelle relative al piano orizzontale x-z. E' possibile assumere che l'oceano sia infinito, risolvendo il tutto tramite integrali di Fourier. Ciò introduce, però, un banale problema di implementazione, cioè l'impossibilità di rappresentare una soluzione "infinita" su un calcolatore che sfrutta l'aritmetica finita. E' pertanto possibile assumere che l'oceano sia descritto da una funzione periodica in *x* e *z* avente *L* come periodo. Tale funzione può essere rappresentata come una serie di Fourier che, pur essendo infinita, è possibile troncare ad un numero finito di componenti. Sia $h(x,y,z,t)$ una componente di Fourier del campo di altezze *h*, definita come segue

$$h(x,z,t) = \hat{h}_{i,j}(t)e^{\frac{\sqrt{-1}2\pi(ix+jz)}{L}}$$

dove $\hat{h}_{i,j}(t)$ è il *coefficiente di Fourier*, il vettore $(i,j)/L$ fornisce la frequenza spaziale e il vettore $\vec{k} = 2\pi(i,j)/L$ è detto *vettore onda*. Sia *k* il *numero di onde* tale che

$$k = \left\|\vec{k}\right\| = \frac{2\pi\sqrt{i^2+j^2}}{L},$$

si definisce *lunghezza d'onda* λ come

$$\lambda = \frac{2\pi}{k} = \frac{L}{\sqrt{i^2+j^2}}.$$

Sarebbe ora necessario svolgere una serie di trasformazioni e passaggi matematici per arrivare alla definizione di parametri utilizzati nella stima delle soluzioni del campo di altezze. Non si vuole, però, rendere la lettura di tale trattazione troppo tediosa, quindi per concentrarsi sugli aspetti implementativi, si preferisce mostrare i risultati di tali trasformazioni senza dilungarsi nelle stesse. Si informa, per completezza, che ciò che viene detto di seguito deriva dalla valutazione del potenziale, nella forma di Fourier analoga a quella del campo di altezze di poc'anzi, tramite le PDE precedenti. Tramite passaggi intermedi è possibile definire la *frequenza delle onde* come

$$\omega_k = \sqrt{kg\frac{1-e^{-2kH}}{1+e^{-2kH}}}.$$

E' possibile inoltre riscrivere la soluzione del campo di altezze, utilizzando una delle soluzioni ottenute dalla valutazione del potenziale, come

$$h(x,z,t) = e^{-\sqrt{-1}\omega_k t} e^{\sqrt{-1}2\pi(ix+jz)/L} = e^{\sqrt{-1}(\vec{k}\cdot(x,z)-\omega_k t)} = e^{\sqrt{-1}\vec{k}\cdot[(x,z)-c_k t\hat{k})]} \quad (4.19)$$

con $\hat{k} = \vec{k}/k$ definito come *direzione unitaria delle onde* e

$$c_k = \frac{\omega_k}{k} = \sqrt{\frac{g(1-e^{-2kH})}{k(1+e^{-2kH})}} \quad (4.20)$$

definita come *velocità delle onde*.

La (4.19) è molto importante, in quanto, fornendo i valori del campo delle altezze, permette di studiare come varia il movimento di un'onda in relazione alla sua velocità nel tempo. La (4.20), invece, definita anche *relazione di dispersione*, fornisce una relazione fra la velocità di un'onda e la sua frequenza d'onda o analogamente il numero di onde. Qui è possibile ricollegarsi alle onde in fisica e, tale relazione indica come la velocità di un'onda sia inversamente proporzionale al numero di onde e direttamente proporzionale alla loro lunghezza d'onda. Quindi, onde di dimensioni differenti, viaggiano a velocità differenti. La (4.20) permette, inoltre, di fare una stima della velocità delle onde sia per acque basse che per oceani, ovvero acque più profonde. Infatti se *H* è molto piccolo allora si ottiene $c \sim \sqrt{gH}$, mentre per valori di *H* grandi $c \approx \sqrt{g/k}$ e $\omega \approx \sqrt{gk}$. Ciò che è stato fatto fino ad ora non è però il caso generale, è servito esclusivamente per ottenere una formula che ci fornisse la stima della velocità delle onde.

La soluzione generale al campo delle altezze è data da

$$h(x,z,t) = \sum_{i,j} A_{i,j} \cos(\vec{k}\cdot(x,z) - \omega_k t + \theta_{i,j}), \quad (4.21)$$

dove $A_{i,j}$ è un valore costante della ampiezza dell'onda, $\vec{k}$ è il vettore onda calcolato come specificato poc'anzi e che punta nella direzione del moto dell'onda stessa, ω_k è la frequenza dell'onda e $\theta_{i,j}$ una costante di sfasamento. La (4.21) può essere utilizzata per ottenere una valutazione di tutti i valori del campo di altezze. Con il tempo, però, sono stati implementati algoritmi efficienti e a basso costo da utilizzare per risolvere trasformate di Fourier o situazioni come quella sopra esposta. Tali implementazioni, ad esempio del *Fast Fourier Transform* (FFT) o *trasformata di Fourier veloce*, sono presenti in numerose librerie all'interno di *Application Programming Interface* (API), conseguentemente, almeno in questa trattazione, se ne fa uso.

Sia $n = 2^m$, si limitino gli indici *i* e *j* in modo che soddisfino $-n/2 + 1 \leq i, j \leq n/2$.

La valutazione della funzione $h(x, z, t)$, relativa al campo di altezze *h*, per un dato istante di tempo *t*, su una griglia regolare *nxn* per le posizioni $0 \leq x, z < L$ è data da

$$h_{pq} = \sum_{i=-\frac{n}{2}+1}^{\frac{n}{2}-1} \sum_{j=-\frac{n}{2}+1}^{\frac{n}{2}-1} A_{i,j} \cos(\vec{k} \cdot (x_p, z_q) - \omega_k t + \theta_{i,j}), \qquad (4.22)$$

dove $x_p = pL/n$ e $z_q = qL/n$. Da questa dopo diversi re-arrangiamenti che in questa sede si omettono si ottiene

$$h_{pq} = \sum_{i=-\frac{n}{2}+1}^{\frac{n}{2}-1} \sum_{j=-\frac{n}{2}+1}^{\frac{n}{2}-1} Y_{i,j}(t) e^{\sqrt{-1}\left(\frac{2\pi(ip+jq)}{n}\right)},$$

dove $Y_{i,j}(\mathrm{t})$ è il coefficiente complesso di Fourier dato da

$$Y_{i,j}(t) = \left[\frac{1}{2}\cos(\theta_{i,j} - \omega_k t) A_{i,j} + \frac{1}{2}\cos(\theta_{-i,-j} - \omega_k t) A_{-i,-j}\right] + \sqrt{-1}\left[\frac{1}{2}\sin(\theta_{i,j} - \omega_k t) A_{i,j} + \frac{1}{2}\sin(\theta_{-i,-j} - \omega_k t) A_{-i,-j}\right]$$

Le implementazioni di FFT permettono di trovare i valori h_{pq} del campo di altezze una volta valutata $Y_{i,j}(t)$. Il vantaggio di utilizzare il metodo finora trattato è dato dal fatto che, una volta ottenuti tutti i valori del campo di altezze, questi possono essere usati direttamente per il rendering. A questo si aggiunge il fatto che, pur avendo la presenza di numeri complessi fin dal primo momento in cui si è utilizzato Fourier ($\sqrt{-1}$), le soluzioni al metodo sopra esposto dovrebbero essere, perturbazioni dell'errore a parte, reali.

4.4.3 Ritorno al caso reale e completamento della tecnica utilizzata

Come si è potuto notare, nel corso della trattazione del metodo per la simulazione degli oceani, si è applicata una serie di semplificazioni che, a livello grafico, rendono inevitabilmente un risultato non realistico. Avendo ristretto la trattazione allo studio del campo di altezze, si è arrivati a modellare un oceano che non possiede altro movimento rispetto a quello verticale. Tuttavia, ϕ possiede la stessa "forma" di *h*, cioè la (4.21) e grazie al suo gradiente può fornire le informazioni sul campo vettoriale, necessario per aggiungere un movimento orizzontale all'oceano. Si ricorda come all'inizio del paragrafo si fosse definito $\nabla\phi = \vec{u}$ (4.17).

Sapendo che

$$\phi(x, y, z, t) = \sum_{i,j} \frac{A_{i,j}\omega_k}{k} \sin(\vec{k} \cdot (x, z) - \omega_k t + \theta_{i,j}) e^{ky}, \qquad (4.23)$$

dove e^{ky} è una delle soluzioni delle PDE, studiate nel corso del paragrafo, allora

$$u(x, y, z, t) = \frac{\partial\phi}{\partial x} = \sum_{i,j} \frac{A_{i,j}\omega_k 2\pi i}{kL} \cos(\vec{k} \cdot (x, z) - \omega_k t + \theta_{i,j}) e^{ky},$$

$$v(x,y,z,t) = \frac{\partial \phi}{\partial y} = \sum_{i,j} A_{i,j}\omega_k \sin(\vec{k}\cdot(x,z) - \omega_k t + \theta_{i,j})e^{ky},$$

$$w(x,y,z,t) = \frac{\partial \phi}{\partial z} = \sum_{i,j} \frac{A_{i,j}\omega_k 2\pi j}{kL} \cos(\vec{k}\cdot(x,z) - \omega_k t + \theta_{i,j})e^{ky}.$$

Di interesse sono le soluzioni dell'equazione $d\vec{x}/dt = \vec{u}(\vec{x}_0, t)$, che descrive lo spostamento dell'acqua da una posizione iniziale $\vec{x}_0$. Il campo generato dagli spostamenti, la cui formula sarà fornita di seguito, permette di gestire anche solidi galleggianti sulla superficie dell'acqua.

$$\Delta x = \sum_{i,j} \frac{-2\pi A_{i,j} i}{kL} \sin(\vec{k}\cdot(x_0, z_0) - \omega_k t + \theta_{i,j})e^{ky_0}$$

$$\Delta y = \sum_{i,j} A_{i,j} \cos(\vec{k}\cdot(x_0, z_0) - \omega_k t + \theta_{i,j})e^{ky_0}$$

$$\Delta z = \sum_{i,j} \frac{-2\pi A_{i,j} j}{kL} \sin(\vec{k}\cdot(x_0, z_0) - \omega_k t + \theta_{i,j})e^{ky_0}$$

Le soluzioni sopra riportate, infatti, costituiscono un campo degli spostamenti che descrive come una particella si sposta, seguendo l'acqua, ad ogni istante di tempo e per qualsiasi valore di $y_0 \leq 0$. Grazie al campo appena creato è possibile, inoltre, avere una rappresentazione della superficie dell'oceano più dettagliata. Se viene sfruttata questa possibilità, il campo degli spostamenti deforma il piano $y = 0$ e, nel caso in cui questo sia applicato ad una componente di Fourier, prende il nome di *onda di Gerstner* [FR86]. Come si è già fatto notare, per completare le soluzioni della (4.23) è necessario introdurre quelle corrispondenti al piano orizzontale analoghe alla (4.22) e a $Y_{i,j}(t)$.

Conseguentemente

$$\Delta x_{pq} = \sum_{i=-\frac{n}{2}+1}^{\frac{n}{2}-1} \sum_{j=-\frac{n}{2}+1}^{\frac{n}{2}-1} X_{i,j}(t) e^{\sqrt{-1}\left(\frac{2\pi(ip+jq)}{n}\right)},$$

$$\Delta z_{pq} = \sum_{i=-\frac{n}{2}+1}^{\frac{n}{2}-1} \sum_{j=-\frac{n}{2}+1}^{\frac{n}{2}-1} Z_{i,j}(t) e^{\sqrt{-1}\left(\frac{2\pi(ip+jq)}{n}\right)},$$

nelle quali

$$\left(X_{i,j}(t), Z_{i,j}(t)\right) = \sqrt{-1}\frac{\vec{k}}{k} Y_{i,j}(t).$$

Queste, analogamente a $Y_{i,j}(t)$ possono essere valutate grazie alla libreria FFT per ottenere il valore delle componenti orizzontali del campo di spostamento valutato sulla griglia.

A ciò che è stato detto sopra, è possibile aggiungere un parametro $\lambda \in (0,1)$ di *"choppiness"* o *"violenza" delle onde.* Questo, permette di modificare lo spostamento

orizzontale, cioè: per $\lambda = 0$ si ottengono le soluzioni del campo di altezze, per $\lambda = 1$ quelle del campo di spostamento. E' poi necessario definire i parametri $A_{i,j}$ e $\theta_{i,j}$ introdotti precedentemente. La costante di sfasamento $\theta_{i,j}$ si può definire come un valore scelto casualmente nell'intervallo $[0,2\pi]$. $A_{i,j}$ va scelta in modo da ottenere una simulazione credibile, infatti se il rapporto tra questa e la lunghezza delle onde è molto alto, allora la superficie tende ad aprirsi, creando buchi. Si può scegliere $A_{i,j} < O(1/k)$. In conclusione si può notare come, per tutto il paragrafo, si sia fatto uso di funzioni sinusoidali che potrebbero rendere visibile ad un occhio attento, un comportamento periodico. Per risolvere questo artefatto sono stati proposti diversi metodi, tuttavia l'argomento è ancora materia di numerosi studi.

Si è così conclusa la trattazione di ambienti particolari costituiti da acqua, quali acque basse e oceani e sono state introdotte tecniche per simulare effetti o proprietà come la viscosità o la turbolenza. Si sono mostrate, quindi, tutte le tecniche necessarie per effettuare una simulazione di fluidi al calcolatore attraverso il metodo di Navier-Stokes. Nei capitoli successivi, saranno mostrati i progetti svolti, volti a mostrare il diretto risultato grafico dell'implementazione delle equazioni di Navier-Stokes e di altri due metodi che verranno poi brevemente discussi: Lattice Boltzmann e SPH.

Capitolo 5

5 Simulazione in Blender 2.76 di un oceano: progetto "Baia"

In questo capitolo si vogliono mostrare le fasi di modellazione, creazione e animazione di un ambiente che simuli il mare. Il progetto è stato svolto grazie all'utilizzo di *Blender 2.76b,* un software *open source* multipiattaforma, sviluppato dalla *Blender foundation,* che permette la modellazione, il "*rigging"*, l'animazione e il rendering di immagini tridimensionali unitamente alla simulazione di fluidi, effetti particellari e corpi deformabili. Blender fornisce diversi metodi per la simulazione di fluidi, siano essi liquidi, come quelli di interesse per questa trattazione, o aeriformi. Ponendo un particolare accento sulle analogie tra i parametri introdotti in questo testo e quelli presenti su Blender, nei paragrafi successivi si mostra come si possa effettuare la simulazione di un oceano o mare. Nei capitoli seguenti, invece, saranno trattati i metodi usati da Blender per la simulazione di fluidi su piccola scala. Di questi, prima di passare ai progetti a loro correlati, si esporrà brevemente la teoria.
Per fare questa prima simulazione, è stato usato il modificatore *Oceano,* offerto da Blender, il quale utilizza il metodo trattato nel paragrafo "Oceani" del capitolo precedente. Il simulatore utilizza i metodi legati alle FFT per generare internamente griglie 2D di dati sulla simulazione. Questi possono riguardare lo spostamento, le normali o dati aggiuntivi utilizzabili per l'aggiunta delle creste delle onde etc.

Il progetto "Baia", nasce per mostrare come sia possibile realizzare simulazioni 3D realistiche di ambienti comprendenti il mare. Per dare una panoramica generale su ciò che è possibile fare e come i parametri, propri del modificatore, influenzino la simulazione, si è voluto creare una sequenza video. La scena si sviluppa a partire da un mare molto mosso, con un temporale in arrivo, mentre una barca si sta dirigendo verso una baia. Durante questo spostamento, il mare si calma e la barca arriva alla baia accompagnata dal bel tempo. Di seguito viene descritta la fase di modellazione dell'ambiente, con particolare attenzione al mare e il successivo rendering.

5.1 Modellazione e richiami alla teoria

Questo progetto si compone di numerosi elementi, che vanno ad arricchire la scena simulata e che, nel seguito, verranno mostrati. Di questi, però, il più importante risulta essere il mare, la cui simulazione è oggetto di questa trattazione. Quando viene applicato ad un oggetto geometrico, il modificatore oceano, nel pannello proprietà è possibile impostare numerosi parametri, diversi dei quali trovano una corrispondenza diretta con la teoria. Di seguito si elencano i più importanti.

- La *geometria* è il più importante e definisce il modo con cui viene simulato il mare ed il suo comportamento. Questa può essere *generate* o *displace*. Utilizzando la geometria *generate*, il simulatore crea delle *tiled mesh* che corrispondono alla risoluzione dei dati e che vanno a sostituire la geometria già esistente, cioè l'oggetto iniziale, con una griglia. Con la seconda geometria, invece, si usa il

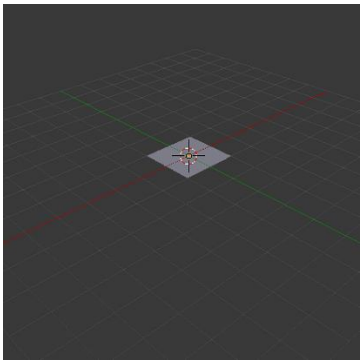


Figura 3a - Piano senza modificatori

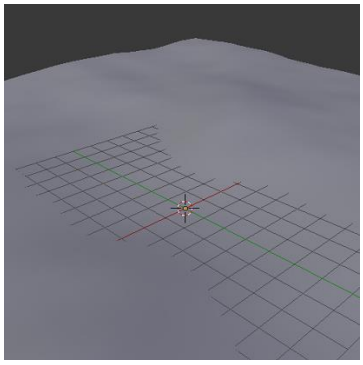


Figura 3b Piano con modificatore Oceano Generate

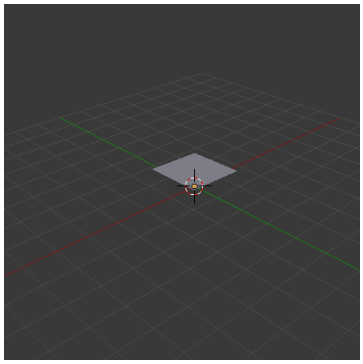


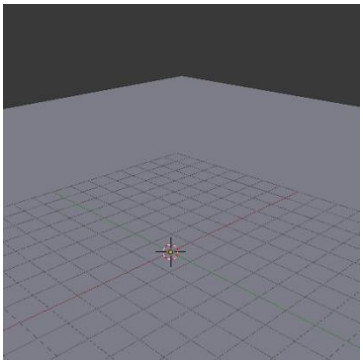
Figura 3c – Piano con modificatore Oceano Displace

piano per la creazione di vertici che vengono poi spostati sull'asse verticale. E' possibile vedere queste due tipologie di geometria nei sotto-paragrafi 4.4.2 e 4.4.3, nei quali la geometria displace corrisponde all'uso del campo di altezze per la simulazione.

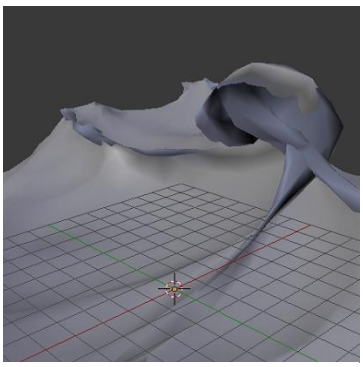
Non essendo state effettuate operazioni di "scalatura", nelle figure 3, 4 e 5 si possono notare alcune differenze che risultano maggiormente visibili nel caso il piano venga animato. Nel caso la geometria sia *generate* l'oggetto iniziale, come già specificato, viene sostituito, conseguentemente è necessario introdurre un canale UV che permetta di far corrispondere allo spazio UV, la griglia simulata.

In questo progetto, per la creazione di un'ambientazione realistica, si è preferito l'utilizzo della geometria *generate* che ha permesso una lavorazione più immediata e semplice.

- Il parametro *tempo* permette, grazie all'impostazione di *keyframe,* di effettuare l'animazione vera e propria del mare.
- Il parametro *depth* o *profondità*, il quale viene trattato nella teoria al paragrafo 4.4 e a cui ci si riferisce con il simbolo *H*, va a modificare la tipologia di mare simulato e la velocità delle onde. Come già specificato nella trattazione, per valori bassi del parametro, si rappresentano acque basse aventi onde più piccole e velocità minore.
- Il parametro *resolution* è quello che va ad influenzare maggiormente la qualità e i costi, di conseguenza la velocità, della simulazione. Questo parametro, infatti, indica la risoluzione della griglia 2D generata dalla simulazione, cioè la dimensione dei dati contenuti in essa, indicata come potenza del due la cui base è il valore del parametro. Una valore pari a quattordici, indica che si avranno dei dati di dimensione $196x196$.

  
  Figura 4 - Oceano con choppiness=0 e scale=0. Caso limite piano.

  Va di per sé che, la qualità della simulazione e il costo computazionale, sono direttamente proporzionali al valore di tale parametro.

  
  Figura 5 - Oceano con choppiness=2 e scale=6. Caso limite mesh rotta.
- *La choppiness* e la *scala* sono due dei parametri più importanti per la definizione delle onde e, quindi, per una loro realistica e plausibile simulazione. Già dal nome, è possibile collegare il primo parametro al capitolo precedente, con la costante λ di choppiness. Come già introdotto (paragrafo 4.4.3), questa permette, assumendo valori nell'intervallo [0,1], di modificare lo spostamento delle onde e i picchi delle onde. Blender pone un limite superiori pari a quattro, tuttavia per valori superiori a uno in combinazione con la scala si possono generare buchi nella mesh. Per choppiness pari a zero, il mare viene spostato lungo l'asse verticale, mentre per valori maggiori, vengono simulati anche gli spostamenti relativi agli assi orizzontali. Il parametro scala, indicato nel paragrafo 4.4.3 come A_{ij}, modifica l'ampiezza delle onde, andando a scalare tutti gli aspetti della simulazione dallo spostamento sugli assi alla schiuma del mare.

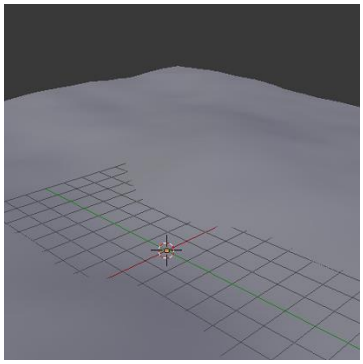


Figura 6a-Choppiness=0 e scale=1.

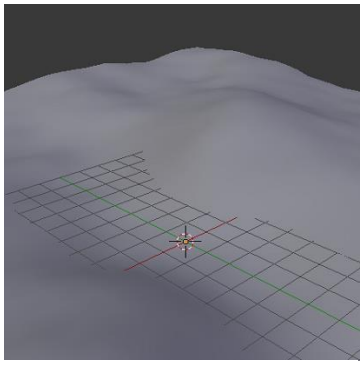


Figura 6b-Choppiness=0 e scale=2.

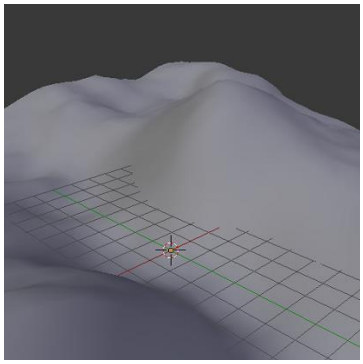


Figura 6c-Choppiness=2 e scale=4.

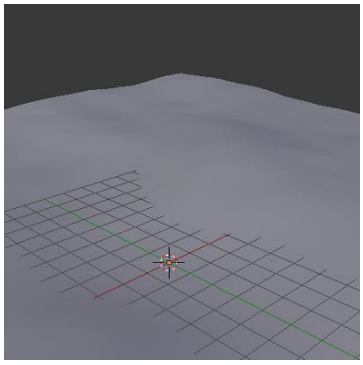


Figura 6d-Choppiness=1 e scale=1

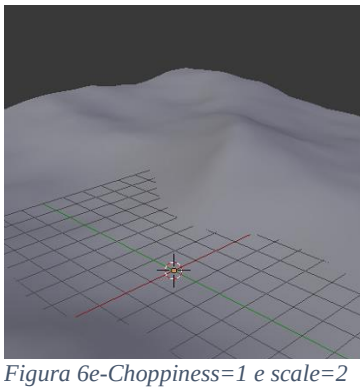


Figura 6e-Choppiness=1 e scale=2

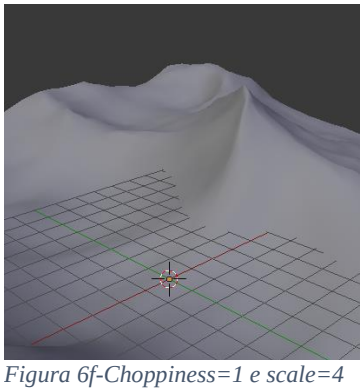


Figura 6f-Choppiness=1 e scale=4

La combinazione di choppiness e scala permette la simulazione di mari relativamente calmi a mari molto mossi. Nelle figure 6 si può comprendere meglio come tali parametri modifichino l'oceano.

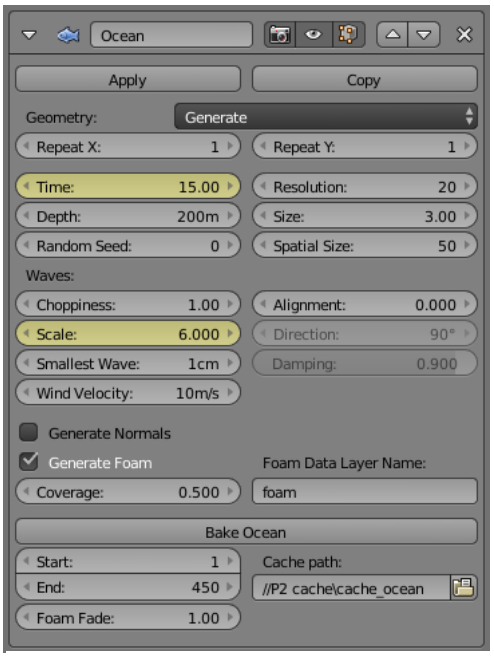


Figura 7 – Pannello dei parametri del modificatore oceano.

- I parametri *allineamento*, *direzione* e *smorzamento*. Il primo specifica la forma delle onde in relazione al vento, per valori elevati, il vento soffia verso una direzione specifica, comprimendo le onde. Il parametro direzione specifica, appunto, tale direzione. Il parametro smorzamento indica, invece, come le onde si riflettono le uno con le altre e la direzione di tale riflessione. E' intuitivo che il secondo e terzo parametro siano subordinati al primo. Infine, vi è la possibilità di impostare un valore minimo limite della dimensione delle onde, applicando così un filtro, un valore di velocità del vento e parametri per la generazione della schiuma.

La figura 7 mostra il pannello proprietà, in cui sono mostrati i valori impostati per i vari parametri nella simulazione dell'oceano.
Considerata la scena che si è voluta creare, i parametri evidenziati di giallo nella figura 7 variano nel tempo, così da poter mostrare gli effetti che questi hanno direttamente a video e ottenere una simulazione realistica.

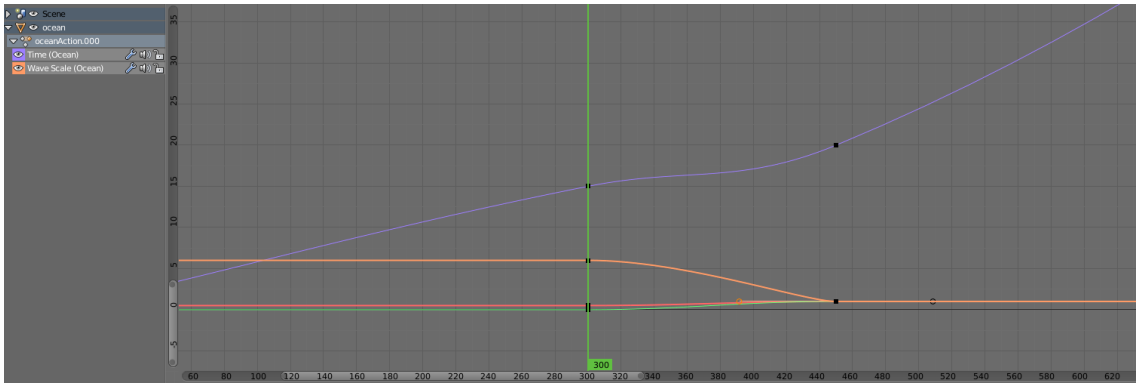


Figura 8 – Graph editor per l'oggetto oceano. In viola il parametro tempo, che controlla l'animazione dell'oceano, in arancione la scala delle onde.

Alla modellazione dell'oceano si unisce quella degli altri elementi della scena, quali la barca, il pontile, le palme, la spiaggia e tutti gli elementi in essa presenti.
Questi, ad eccezione delle palme e dei cespugli, sono stati modellati a partire da forme geometriche di base tramite le tecniche fornite da Blender. Le palme, infatti, data la complessità di modellazione di un albero, sono state realizzate grazie ad un plug-in.
Di seguito si mostra il modello dell'intera spiaggia.

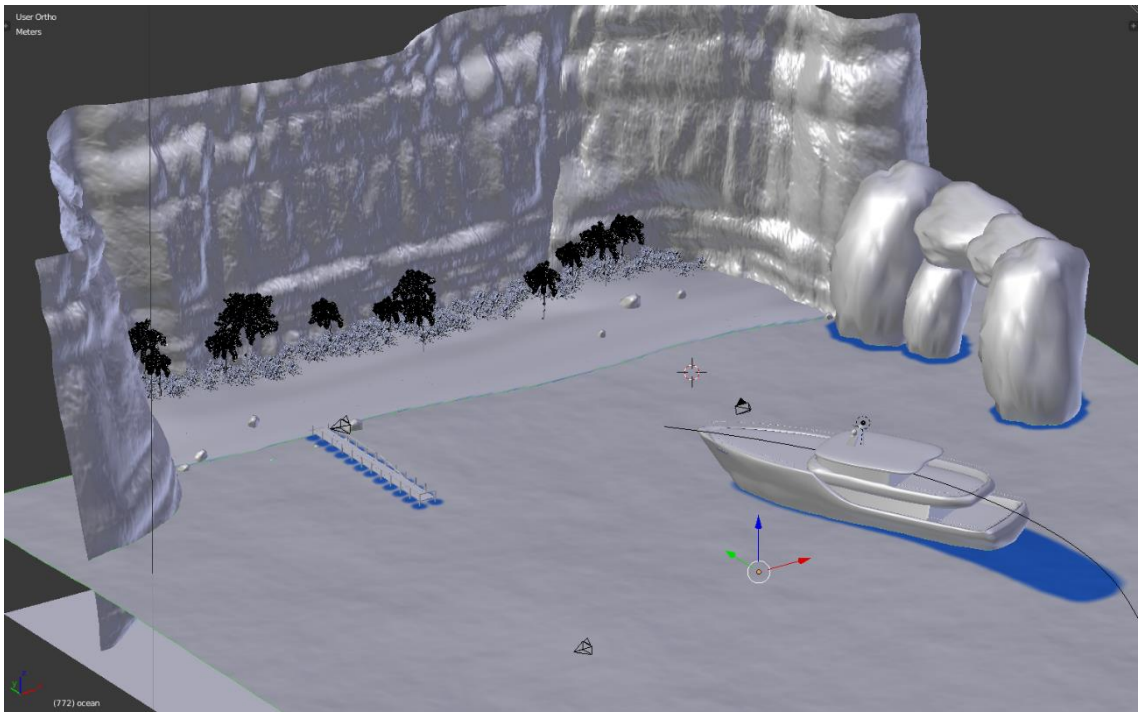


Figura 9-Modello baia. Comprendente: il mare, la barca, il pontile, i cespugli, le rocce, le conchiglie e le palme.

5.2 Materiali e rendering

Una volta effettuata la modellazione tridimensionale dei vari oggetti presenti nella scena, si sono aggiunti i materiali necessari per aggiungere caratteristiche grafiche, come il colore, gli *shader*, le *texture* etc., visualizzabili una volta eseguito il rendering. Per tutti gli elementi è stato usato il *node editor*, un potente strumento che permette, in maniera intuitiva, tramite l'unione di nodi rappresentanti proprietà grafiche di assegnare un materiale ad un oggetto. Di seguito si mostra quello del mare.

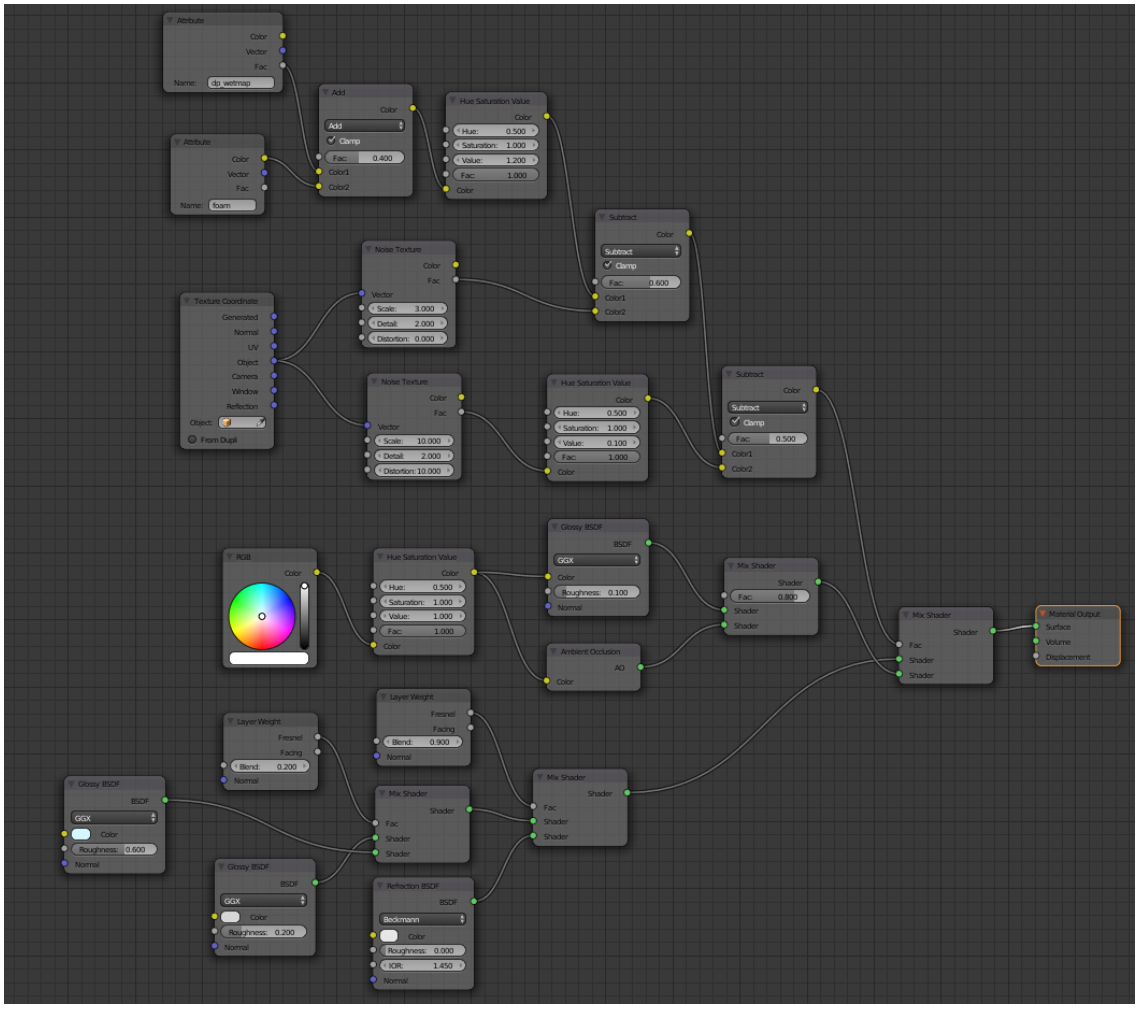


Figura 10 – Materiali dell'oceano mostrati nel Node Editor.

Grazie all'utilizzo del node editor è stato possibile realizzare un materiale realistico per la schiuma del mare e quella prodotta dalla barca, dal pontile e dalle rocce.

Può essere di interesse sapere che, per creare la schiuma della barca e degli altri oggetti della scena, così come l'effetto "bagnato" sulla spiaggia, si è utilizzata la tecnica detta *dynamic painting*. Questa permette di creare effetti, colorare, modificare in modo dinamico un materiale a partire da un altro oggetto della scena.

Di seguito si mostra il rendering del mare, in modo da far capire come sia possibile ottenere la simulazione di quest'ultimo, combinando una buona modellazione alle potenzialità di Blender.

Figura 11 – Oceano mosso renderizzato.

5.3 Confronto sui tempi di rendering

In questo paragrafo, si vuole dare qualche dato sui tempi di rendering, in relazione all'aspetto che maggiormente influenza i costi computazionali, ovvero la risoluzione della griglia. Il confronto viene fatto sulla scena iniziale della simulazione, in quanto, aggiungendo oggetti a quest'ultima, inevitabilmente il tempo richiesto aumenta. E' inoltre necessario specificare che il rendering è eseguito con la GPU, una NVIDIA GeForce GTX 850M e che, quindi, i tempi potrebbero variare con l'utilizzo di schede grafiche differenti o della CPU.

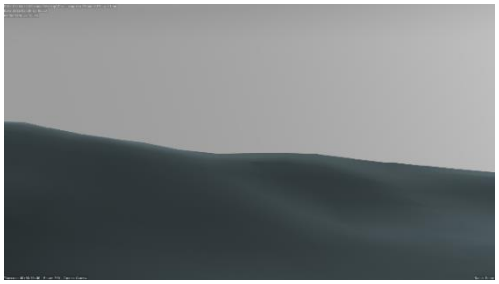


Figura 12a – Oceano, risoluzione=5, rendering time=02:55.45

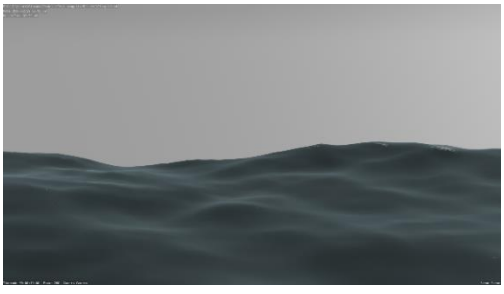


Figura 12b – Oceano, risoluzione=10, rendering time=02:59.46

Figura 12c – Oceano, risoluzione=20, rendering time=03:11.48

Figura 12d – Oceano, risoluzione=30, rendering time=03:36.58

5.4 Risultato finale

Grazie alla conoscenza della teoria alla base della simulazione dei fluidi, unitamente alle potenzialità di Blender, è stato possibile realizzare una sequenza video che mostrasse in modo realistico la scena descritta all'inizio del capitolo. Di seguito vengono riportati degli *screenshot* che mostrano il risultato finale.

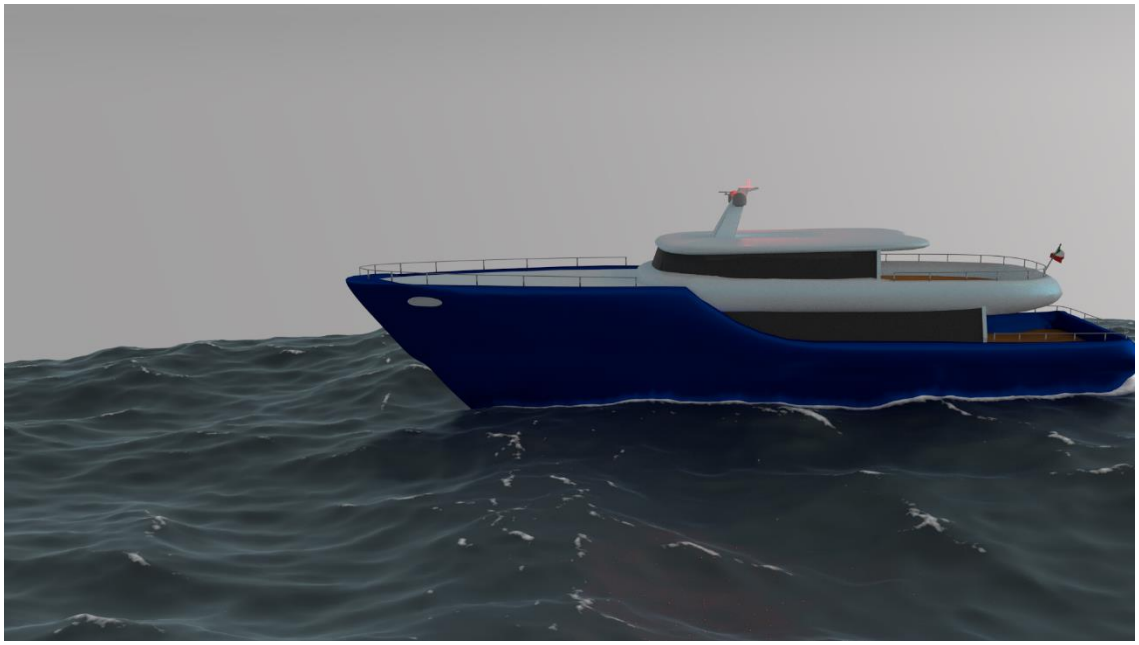


Figura 13a – Mare mosso con barca

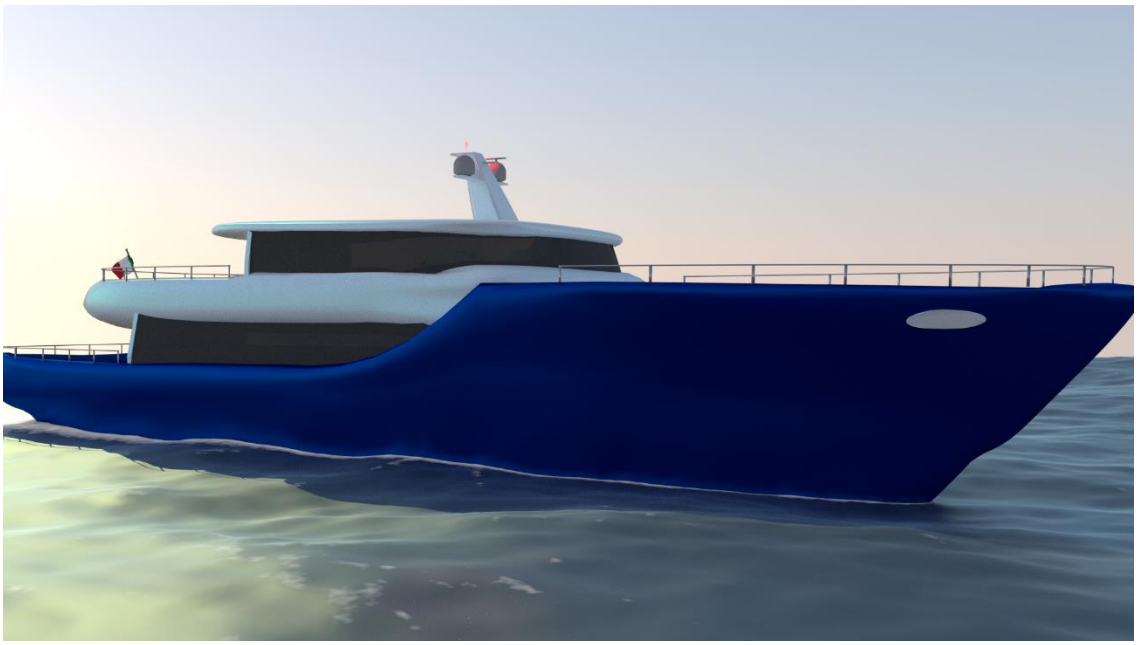


Figura 13b – Mare calmo con barca

Figura 13c – Baia

Capitolo 6

6 Altri metodi per la simulazione di fluidi

Fin dall'inizio della trattazione, il metodo proposto per la simulazione di fluidi al calcolatore, si è basato sull'implementazione di un algoritmo che risolvesse le equazioni di Navier-Stokes. Tuttavia, tale metodo essenzialmente euleriano, è solo uno dei modi con cui è possibile effettuare la simulazione. Nel seguente capitolo sono trattati due metodi non euleriani che simulano i fluidi: il metodo *Lattice Boltzmann* (LB) o *metodo reticolare di Boltzmann*, proposto nei primi paragrafi, si basa sulla risoluzione dell'equazione di Boltzmann mediante modelli di collisione di particelle, mentre il metodo *Smoothed-Particle Hydrodynamics* (SPH), nei paragrafi successivi, fa uso di un sistema particellare. Dalla distinzione di questi due metodi, si può affermare che il metodo LB è una via di mezzo tra i metodi puramente euleriani e quelli lagrangiani, mentre il metodo SPH è totalmente lagrangiano. Siccome l'obbiettivo di questa tesi è di esporre in modo dettagliato il metodo Navier-Stokes, in questo capitolo si cercherà di fornire al lettore solo le conoscenze necessarie per comprendere e sfruttare i metodi LB e SPH senza, però, entrare nei dettagli. A seguito della trattazione di ciascun metodo, si mostrerà un progetto, fatto utilizzando Blender 2.76b, che utilizza il metodo esposto in una particolare simulazione, la quale metterà a fuoco un aspetto, trattato nel capitolo 4, non ancora simulato.

6.1 Metodo Lattice Boltzmann

Il metodo Lattice Boltzmann, basato sulla teoria cinematica dei gas di Ludwig Boltzmann, considera il fluido come un insieme di tante piccole particelle fittizie che si muovono in modo casuale. Secondo tale modello, lo scambio di energia e di quantità di moto che avviene tra le particelle si ottiene dalla propagazione e collisione di quest'ultime. Questo scambio viene descritto *dall'equazione del trasporto di Boltzmann*

$$\frac{\partial f}{\partial t} + \vec{u} \cdot \nabla f = \Omega, \tag{6.1}$$

dove $f(\vec{x},t)$ è la funzione di distribuzione di una particella, $\vec{u}$ la sua velocità e Ω un operatore di collisione. L'idea alla base del metodo LB, è quella di semplificare la dinamica dei fluidi di Boltzmann riducendo il numero di particelle, racchiudendole poi nei nodi di un reticolo (*lattice*). Questi reticoli possono essere di diverso tipo, sia cubici che triangolari, producendo uno svariato numero d metodi diversi. Per classificare tali metodi si fa quindi uso dello schema *DnQm*, secondo il quale *n* indica il numero di dimensioni mentre *m* il numero di velocità. Conseguentemente, il modello che useremo noi, definito *D2Q9*, sarà bidimensionale e le particelle potranno propagarsi in nove direzioni, incluso lo stato di quiete, con altrettante velocità. Utilizzando tale modello, siano $\vec{e}_i$ con $i = 0,\dots,8$ le *velocità microscopiche*, così definite

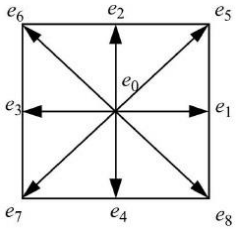


Figura 14- Cella D2Q9

$$\vec{e}_i = \begin{cases} (0,0) & i = 0 \\ (1,0),(0,1),(-1,0),(0,-1) & i = 1,2,3,4 \\ (1,1),(-1,1),(-1,-1),(1,-1) & i = 5,6,7,8 \end{cases}$$

Ad ogni particella sul reticolo è poi associata la funzione di distribuzione di probabilità discreta poc'anzi introdotta f come $f_i(\vec{x},\vec{e}_i,t)$ o $f_i(\vec{x},t)$ per $i = 0,\dots,8$ che descrive la probabilità con cui la particella stessa può propagarsi in una determinata direzione. Sia ρ la *densità macroscopica del fluido*, questa è definita come

$$\rho(\vec{x},t) = \sum_{i=0}^{8} f_i(\vec{x},t), \tag{6.2}$$

mentre la *velocità macroscopica* $\vec{u}(\vec{x},t)$ viene definita come la media delle velocità microscopiche, sopra definite, pesata dalla funzione di distribuzione f_i, cioè

$$\vec{u}(\vec{x},t) = \frac{1}{\rho}\sum_{i=0}^{8} c f_i \vec{e}_i. \tag{6.3}$$

Il metodo LB, una volta definite tali quantità, procede risolvendo l'equazione ottenuta eguagliando la propagazione

$$f_i(\vec{x} + c\vec{e}_i\Delta t, t + \Delta t) - f_i(\vec{x},t),$$

alla collisione

$$-\frac{|f_i(\vec{x},t) - f_i^{eq}(\vec{x},t)|}{\tau},$$

ottenendo

$$f_i(\vec{x} + c\vec{e}_i\Delta t, t + \Delta t) - f_i(\vec{x},t) = -\frac{|f_i(\vec{x},t) - f_i^{eq}(\vec{x},t)|}{\tau}, \tag{6.4}$$

dove $f_i^{eq}(\vec{x},t)$ è la funzione di distribuzione di equilibrio del metodo di collisione e τ il tempo di rilassamento verso l'equilibrio.

Prima di procedere, è necessario scegliere un modello di collisione, dal quale dipende la f_i^{eq}. Comunemente si utilizzano le *collisioni di Bhatnagar-Gross-Krook* (BGK), per le quali la f_i^{eq} è definita come

$$f_i^{eq}(\vec{x}, t) = w_i \rho + \rho s_i(\vec{u}(\vec{x}, t)), \tag{6.5}$$

dove $s_i(\vec{u})$ è data da

$$s_i(\vec{u}) = w_i \left[3 \frac{\vec{e}_i \cdot \vec{u}}{c} + \frac{9}{2} \frac{(\vec{e}_i \cdot \vec{u})^2}{c^2} - \frac{3}{2} \frac{\vec{u} \cdot \vec{u}}{c^2} \right],$$

w_i sono i pesi

$$w_i = \begin{cases} 4/9 & i = 0 \\ 1/9 & i = 1,2,3,4 \\ 1/36 & i = 5,6,7,8 \end{cases}$$

e $c = \Delta x / \Delta t$ è la velocità del reticolo.

Infine, il coefficiente di viscosità cinematica ν è definito come

$$\nu = \frac{2\tau - 1}{6} \frac{(\Delta x)^2}{\Delta t}. \tag{6.6}$$

Come per il metodo di Navier-Stokes anche per il metodo LB è necessario porre delle condizioni ai bordi. Non volendosi dilungare troppo sul metodo proposto in questo capitolo, si accenna alle due principali condizioni ai bordi imponibili, per poi presentare l'algoritmo alla base di LB.

Una prima condizione ai bordi molto utilizzata è quella definita *bounce-back* che implementa l'ormai nota condizione no-slip. L'idea del bounce-back è quella di rispedire indietro le particelle che raggiungono il bordo con una data funzione di distribuzione discreta, lungo la direzione con la quale sono arrivate. Questo tipo di condizione può avere l'implementazione *on-grid* per la quale il bordo del dominio del fluido è allineato ai punti del reticolo, oppure l'implementazione *mid-grid* nella quale il bordo del dominio è compreso tra i nodi del reticolo e nuovi nodi fittizi di quest'ultimo. La principale differenza tra queste due implementazioni è l'accuratezza, nel primo caso di prim'ordine, nel secondo caso di second'ordine.

La seconda condizione ai bordi, introdotta da Zou e He, si basa sul voler mantenere un certo valore di pressione e velocità ai bordi e per fare ciò risolve un sistema lineare. Il problema insito in questa seconda condizione ai bordi, tuttavia, è la sua dipendenza dall'orientazione del bordo, e quindi, a differenza della prima condizione, la sua generalizzazione per geometrie complesse potrebbe essere difficile.

Di seguito si mostra l'algoritmo di LB per la simulazione di fluidi al calcolatore:

1. Si impostino i valori di ρ, $\vec{u}$, f_i e f_i^{eq}.
2. Si svolga l'operazione di propagazione nella direzione di $\vec{e}_i$, ottenendo f_i^*.
3. Si calcoli il valore macroscopico della densità ρ e della velocità $\vec{u}$, utilizzando la (6.2) e (6.3).
4. Si calcoli f_i^{eq} utilizzando la (6.5)
5. Si svolga l'operazione di collisione, calcolando le funzioni di distribuzione aggiornate $f_i = f_i^* - \frac{1}{\tau}(f_i^* - f_i^{eq})$, utilizzando la (6.4).
6. Si ripetano gli step dal 2 al 5.

6.2 Simulazione in Blender di fluidi diversi: progetto "Vasche"

Blender 2.76b fornisce un simulatore di fluidi che implementa il metodo LB trattato nel paragrafo precedente. Si è deciso, quindi, di creare una sequenza video nella quale si potesse vedere il risultato ottenuto con questa tecnica. Il simulatore di fluidi di Blender permette di creare scene realistiche, dando l'opportunità di aggiungere alla simulazione un effetto molto importante come la viscosità. L'utilizzo di tali funzionalità e, quindi, del metodo LB permette di rappresentare, al calcolatore, fluidi su piccola scala, come vasche d'acqua, spruzzi, pozze ma anche scene animate come fontane, dighe etc.

Il progetto "Vasche" mira a mostrare le differenze tra tre tipologie di fluido, diverse, appunto, per viscosità: l'acqua, della vernice e del vetro fuso.

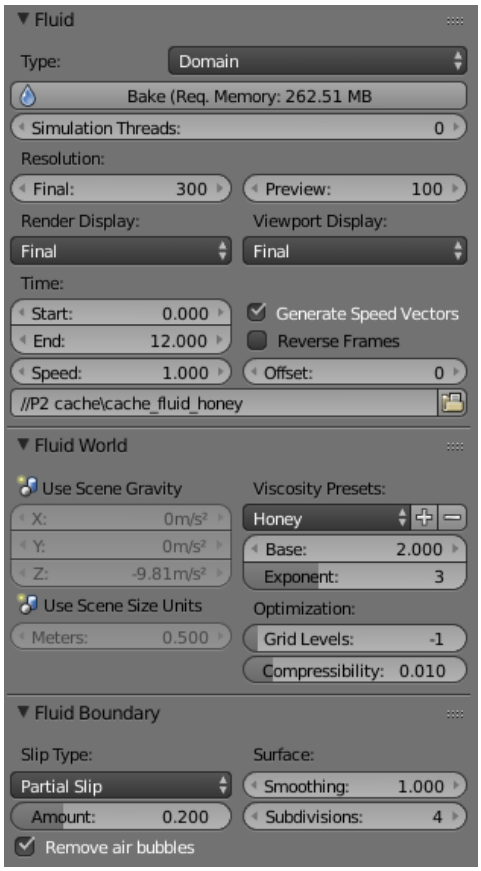


Figura 15 – Pannello delle proprietà del dominio di un fluido.

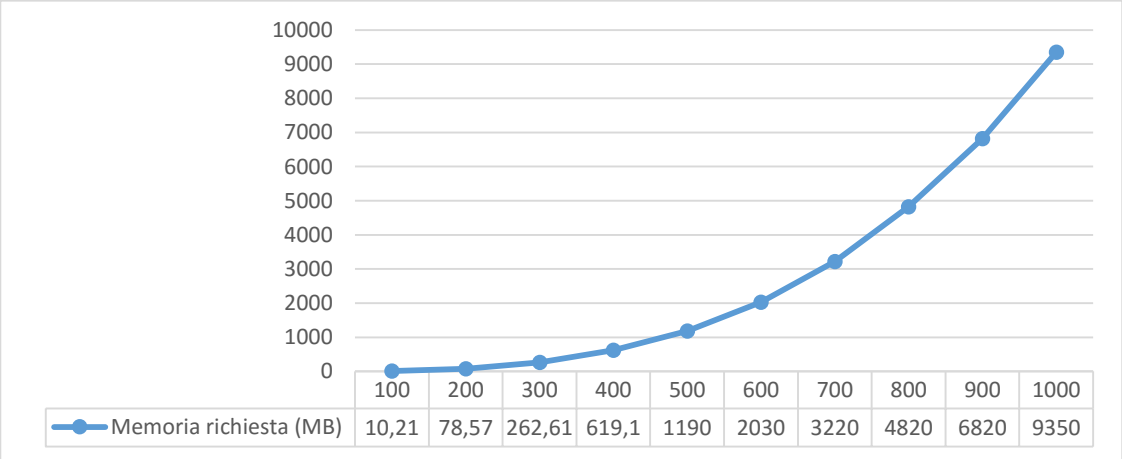
6.2.1 Modellazione e richiami alla teoria

Per mostrare tali differenze, la scena è stata mantenuta molto semplice e la modellazione ha riguardato la vasca, gli ostacoli presenti in essa per evidenziare l'aspetto viscoso dei fluidi e i fluidi stessi. Il simulatore di Blender permette di definire: il *dominio* della simulazione, *l'oggetto fluido*, *gli ostacoli*, *oggetti di immissione* e *rimozione di fluido*, oggetti per controllare la simulazione e oggetto

particelle. Di questi il più importante è il dominio, introdotto nel paragrafo precedente, descrive l'area nella quale avviene la simulazione. Nel dominio è possibile impostare la quasi totalità dei parametri di interesse, perciò ci concentreremo su di esso.

- La *final resolution* è il parametro che indica la granularità con la quale la simulazione del fluido viene eseguita. Determinando l'ammontare dei dettagli all'interno di quest'ultima, la risoluzione è il parametro che influisce maggiormente sulla memoria utilizzata e il tempo computazionale.

  Nella figura 15, si forniscono i dati che mostrano la relazione tra la memoria utilizzata e la risoluzione. Si approfitta dei risultati sopra riportati per motivare la scelta della scena. Per poter ottenere un buon risultato, senza gravare troppo sulla CPU, si è voluto modellare pochi oggetti con materiali semplici, scegliendo come valore di risoluzione del fluido, 300. E' necessario far presente che valori molto alti, cioè maggiori di 500, tendono a rendere irrealistica la simulazione.



	100	200	300	400	500	600	700	800	900	1000
Memoria richiesta (MB)	10,21	78,57	262,61	619,1	1190	2030	3220	4820	6820	9350

Figura 16 – Tabella in cui si nota la relazione tra la risoluzione del dominio e la memoria richiesta per il bake.

- La *viscosità* è trattata nel capitolo quattro molto approfonditamente. Avendo questo progetto lo scopo di mettere in relazione fluidi con viscosità differente, nel proseguo saranno mostrati i risultati ottenuti. Va specificato che Blender utilizza la viscosità cinematica, differente da quella a cui si è soliti riferirsi, in quanto ottenuta da quest'ultima dopo averla divisa per la densità.
- I parametri relativi ai *bordi del dominio* e alla *superficie* si trovano nella sezione *"Fluid boundary"* del pannello in figura 14. Blender mette a disposizione tre opzioni di scorrimento: *no-slip* che non permette il movimento del fluido sulla superficie, *free-slip* e *part-slip* che, invece, lo consentono. Le suddette condizioni possono essere ritrovate nella teoria nei paragrafo 1.4, 2.4. I parametri relativi alla

superficie, riguardano lo *smoothing* applicato ad essa e il *numero di suddivisioni* di un voxel di fluido, permettendo la creazione di mesh ad alta risoluzione dinamicamente.

Per completare la modellazione del progetto, va specificato che, si è scelto di impostare una velocità iniziale ai fluidi in modo da dotarli di un'animazione più complessa, che permettesse di far vedere l'effetto prodotto al contatto con gli ostacoli. Di seguito sono mostrate alcune immagini esemplative della modellazione di tale progetto.

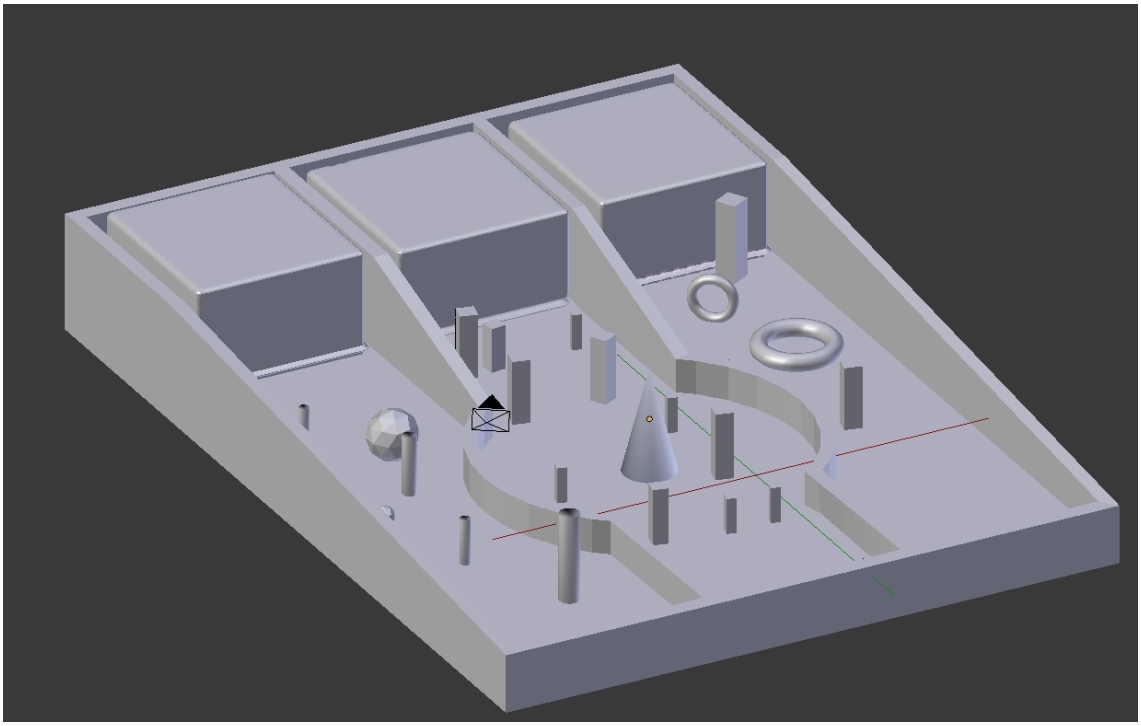


Figura 17 – Modello del progetto "Vasche" all'inizio dell'animazione

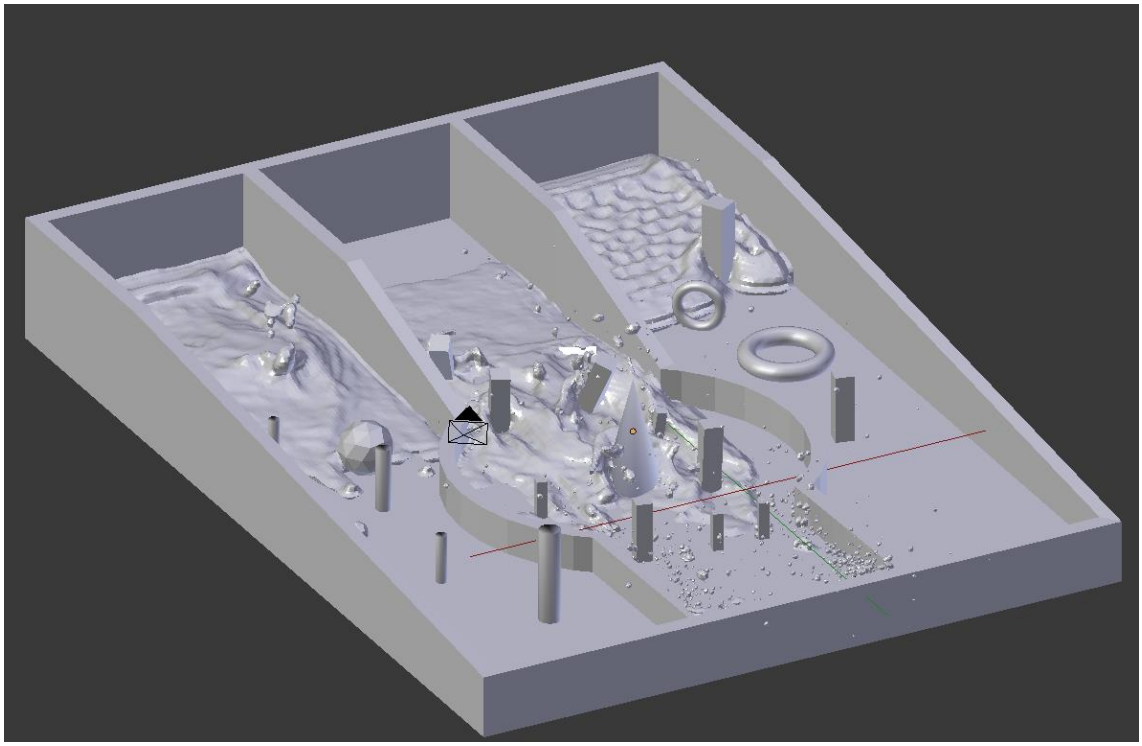


Figura 17a – Modello del progetto "Vasche" alla metà dell'animazione

6.2.2 Materiali e rendering

Anche per gli oggetti presenti in questo progetto è stato utilizzato il node editor, seppur la complessità della scena non richiedesse materiali particolari. Sono stati creati quindi tre materiali per i fluidi, uno *glass* per l'acqua, uno giallo intenso molto diffusivo per il vetro fuso e uno giallo-ocra per la vernice.

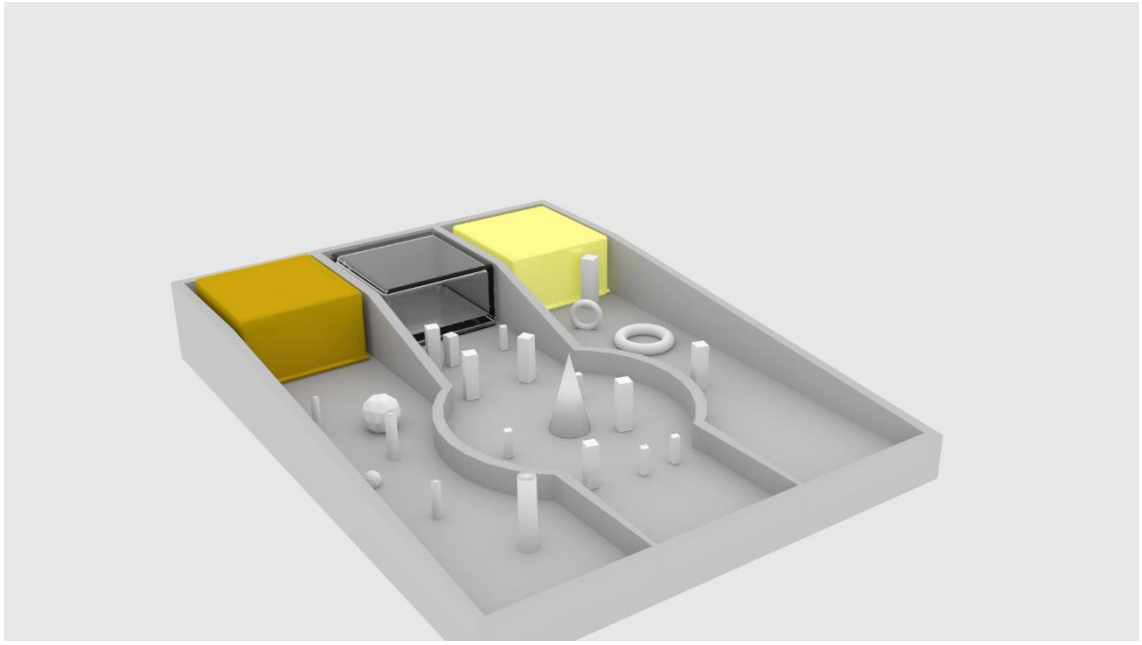


Figura 18a – Progetto "Vasche" all'inizio dell'animazione renderizzato.

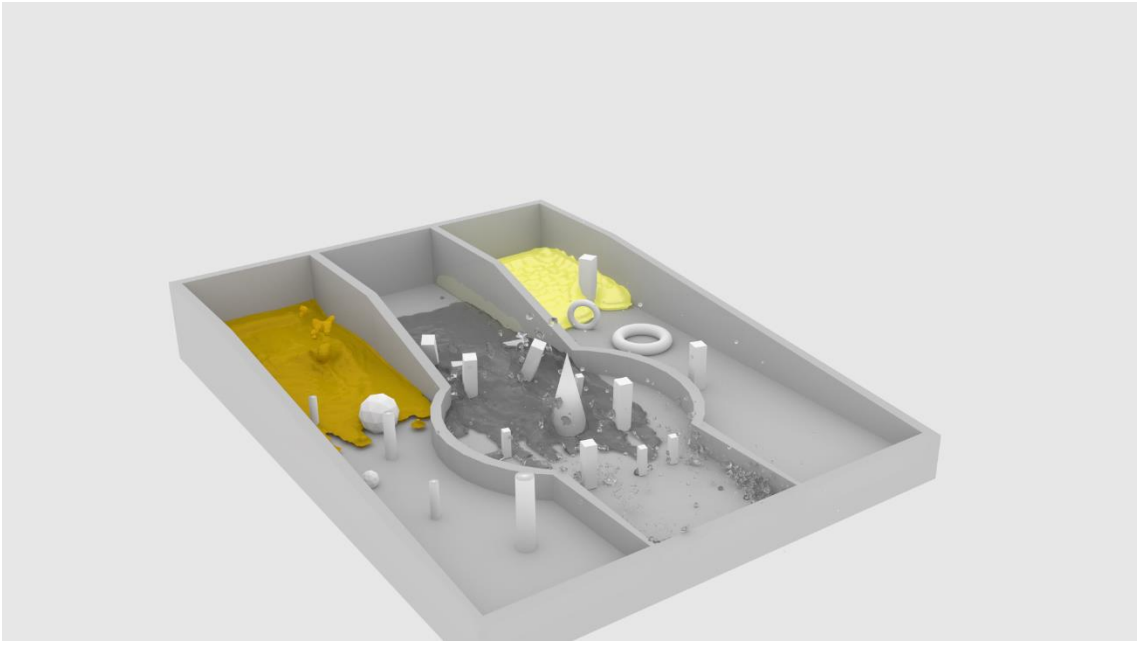


Figura 18b – Progetto "Vasche" alla metà dell'animazione renderizzato.

6.3 Metodo Smoothed Particle Hydrodynamics

Il metodo SPH è stato sviluppato da J. J. Monaghan [MON92] per l'uso in campo astrofisico, balistico, oceanografico e vulcanologico. In linea con l'approccio lagrangiano, di cui fa parte, questo metodo divide il fluido in un sistema particellare, dove ogni particella possiede una massa *m*, una posizione *x*, una velocità *u* e una densità ρ, dipendente dalle densità delle particelle vicine. L'idea alla base di SPH è quella di utilizzare una *funzione kernel* per fare lo smoothing del valore delle proprietà di una particella sulla base della distanza spaziale, solitamente definita *lunghezza di smoothing h*, dalle altre. Conseguentemente, solo le proprietà delle particelle comprese nel raggio della suddetta funzione influenzeranno il valore della proprietà della particella di interesse. Sia *W* la funzione kernel, questa potrebbe essere la funzione Gaussiana oppure una spline cubica. L'utilizzo di quest'ultima permette, ad esempio, di ignorare le particelle più lontane da quella di interesse, diminuendo notevolmente il costo computazionale dell'intero algoritmo. Sia *Q* una quantità, allora la funzione che valuta quest'ultima nel punto *x*, è definita come

$$A(x) = \sum_i m_i \frac{A_i}{\rho_i} W(|x - x_i|, h),$$

dove *i*, indica la particella e ρ_i la sua densità calcolabile, come detto prima grazie alla funzione kernel *W* come

$$\rho_i = \rho(x_i) = \sum_j m_j \frac{\rho_j}{\rho_j} W\big(|x_i - x_j|, h\big) = \sum_j m_j\, W\big(|x_i - x_j|, h\big).$$

Di semplice e immediato calcolo il gradiente di *A*, cioè come varia questa nel tempo, ovvero

$$\nabla A(x) = \sum_i m_i \frac{A_i}{\rho_i} \nabla W(|x - x_i|, h).$$

Le equazioni trovate poc'anzi permetto anche di trovare una versione della legge di conservazione idrodinamica, così da produrre semplici equazioni del moto per le particelle. Sia P_i la pressione, di cui si possiede il valore, per ogni particella *i*, allora la forza esercitata da P_i può essere espressa come

$$F_i = -m_i \sum_{j \neq i} m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2}\right) \nabla_i W_h^{ij},$$

dove

$$\nabla_i W_h^{ij} = \nabla\big(|x_i - x_j|, h\big).$$

Per tenere conto della viscosità, invece, è necessario aggiungere una forza di smorzamento definita come

$$D_i = -m_i \sum_{j \neq i} m_j \Pi_{ij}\, \nabla_i W_h^{ij},$$

dove

$$\Pi_{ij} = \begin{cases} \dfrac{-c\mu_{ij} + 2\mu_{ij}^2}{\bar{\rho}_{ij}} & se\ \mu_{ij} < 0 \\ 0 & se\ \mu_{ij} \geq 0 \end{cases}$$

$$\mu_{ij} = h \frac{\vec{u}_{ij} \cdot \vec{x}_{ij}}{\vec{x}_{ij}^2 + h^2/100},$$

$$\vec{u}_{ij} = \vec{u}_i - \vec{u}_j,$$

$$\vec{x}_{ij} = \vec{x}_i - \vec{x}_j,$$

$$\bar{\rho}_{ij} = (\rho_i - \rho_j)/2,$$

e *c* è la *velocità del suono del fluido simulato*, cioè la velocità più elevata di un'onda che si propaga. Grazie a tutti questi elementi è possibile descrivere completamente un fluido composto da particelle.

SPH possiede numerosi vantaggi, tra cui la garanzia che la massa del fluido simulato si conservi senza alcun costo computazionale aggiuntivo. Inoltre, calcola la pressione attraverso una media pesata dei valori delle particelle vicine, evitando di risolvere un sistema lineare. Analogamente ad altri metodi, però, come si è visto nei capitoli precedenti, necessita di tecniche per la creazione di superfici geometriche renderizzabili.

6.4 Simulazione in Blender di fluidi mediante particelle: progetto "Tazze"

Il progetto "Tazze" mostra, in Blender 2.76b, l'utilizzo del metodo SPH accennato nel paragrafo precedente. SPH, in quanto metodo lagrangiano, prevede l'utilizzo di un elevato numero di particelle per la simulazione di un fluido al calcolatore e ciò si può riscontrare nel fatto che, la sua implementazione in Blender 2.76b, rientra sotto la categoria degli effetti particellari.

Il risultato finale viene reso da un insieme di fattori e di parametri mostrati in figura 19.

- Il *numero di particelle* e la loro *dimensione* sono due parametri molto importanti. La prima fra queste due grandezze, infatti, è quella che influisce maggiormente sul costo computazionale e conseguentemente, sul tempo di simulazione e rendering. La scelta di creare una scena semplice, quindi, è stata subordinata al non voler appesantire troppo la fase di baking e rendering, aumentando esponenzialmente la sua durata.
- La *velocità* dell'emettitore permette di assegnare alle particelle una velocità iniziale la quale sarà poi modificata secondo le leggi fisiche e aggiornata come esposto nel paragrafo precedente. E' possibile, inoltre, impostare un valore per le *forze* che agiscono su di esse.

Figura 19-Pannello delle proprietà di un sistema particellare.

- I parametri *drag, damp, stiffness, viscosity, buoyancy* e *interaction radius* fanno tutti parte della sezione *"Physics-Fluids"*. Il parametro drag, ovvero trascinare, rallenta l'emissione delle particelle in relazione alla loro velocità e dimensione, permettendo alle stesse di essere maggiormente affette da altre proprietà. Il parametro damp, invece, riduce la velocità in generale. La stiffness regola

l'incomprimibilità del fluido, il parametro viscosity la sua viscosità, mentre buoyancy rappresenta una forza nella direzione opposta a quella gravitazionale, che agisce in relazione alla differenza di pressione nel fluido. Infine, l'interaction radius rappresenta il raggio entro cui le particelle del fluido interagiscono tra di loro.

- *Metodo di integrazione*. Blender 2.76b permette anche di scegliere quest'ultimo utilizzato per calcolare la nuova posizione delle particelle. Questo è probabilmente il parametro più importante, in quanto, da come si è potuto notare leggendo la trattazione, il metodo di integrazione produce inevitabilmente diffusione numerica (Paragrafo 2.2). Ciò può portare a comportamenti errati nella simulazione, facendola perdere di realismo. Blender mette a disposizione quattro metodi di integrazione, *RK4*, *Eulero in avanti*, *punto medio* o RK2 e *Varlet* ognuno dei quali possiede pregi e difetti, già trattati nel corso di questo documento. Dei quattro metodi implementati si può affermare che, la tecnica di interazione che offre i migliori risultati è RK4, pur essendo molto lenta, mentre Eulero in avanti risulta essere molto veloce ma poco accurato. Una buona tecnica, che offre discreti risultati nella quasi totalità dei casi, senza inficiare pesantemente sul tempo di esecuzione è quella del punto medio o RK2. In alternativa a questa, la quarta tecnica proposta, Varlet, risulta essere anch'essa molto stabile, veloce e, insieme a RK2 e RK4, conserva discretamente l'energia delle particelle.

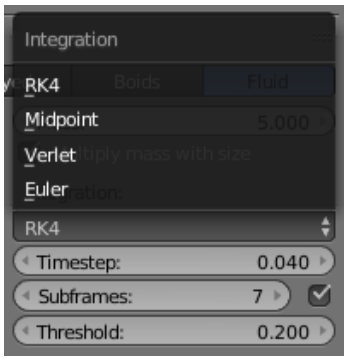


Figura 20- Pannello scelta metodo di integrazione.

- Il pannello *emission* permette, infine, di scegliere oltre che il numero e la *vita* delle particelle anche come queste vengono emesse. La teoria relativa all'emissione delle particelle si può trovare nel paragrafo 3.1.

Come per i due progetti esposti in questa trattazione, anche per questo, l'aggiunta dei materiali agli oggetti della scena è stata effettuata tramite node editor. Come metodo di integrazione per ottenere buoni risultati è stato scelto RK4.

Nella scena sono presenti due tazze, create a partire dalla modellazione di solidi tridimensionali e due piani che hanno il compito di emettere le particelle nell'ambiente.

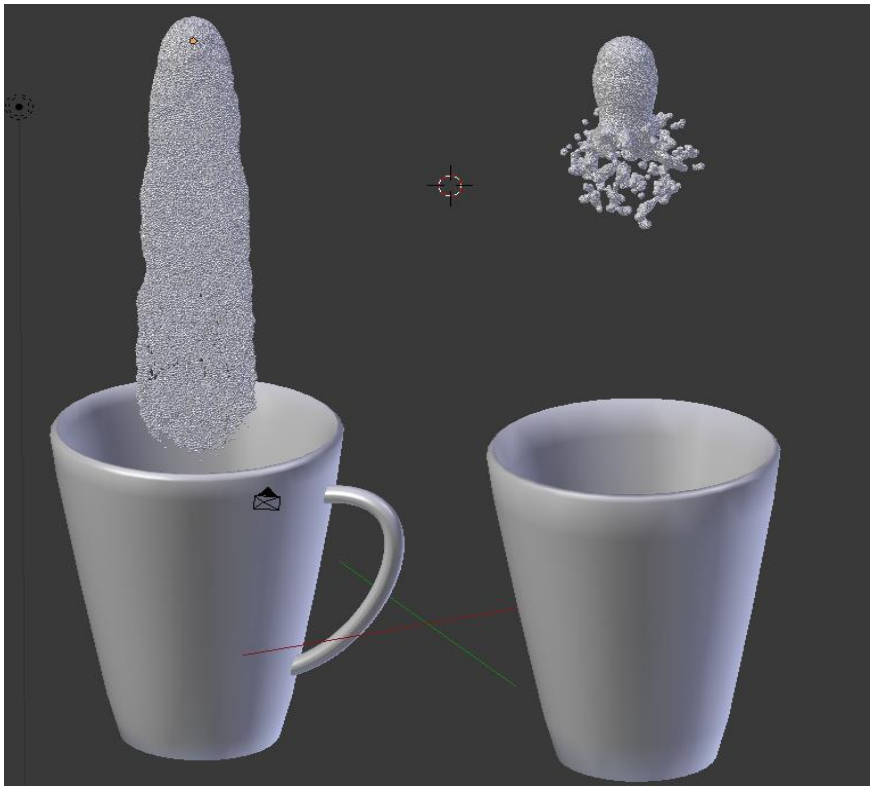


Figura 21 – Modello del progetto "Tazze"

Figura 22 – Progetto "Tazze" renderizzato.

Conclusioni

Dopo aver trattato tre metodi differenti per la simulazione di fluidi, l'euleriano Navier-Stokes, il semi-lagrangiano Lattice Boltzmann e il lagrangiano Smoothed Particles Hydrodynamics è possibile fare una riflessione, già accennata nel corso della trattazione, sui casi di utilizzo. Il primo metodo, che implementa direttamente un risolutore delle equazioni di Navier-Stokes, permette la simulazione realistica di tutti i fluidi, anche differenti per proprietà. Questo, tuttavia, risulta molto complesso a meno di notevoli semplificazioni sul modello del sistema da trattare. Il metodo di Lattice Boltzmann al contrario, rappresenta una buona via di mezzo, permettendo di ottenere risultati ottimi su fluidi, però, di piccola scala. Infine il SPH rappresenta una "particolare" alternativa ai primi due. Per ottenere, infatti, una simulazione altamente realistica di un fluido, serve un numero di particelle molto spesso proibitivo per i tipici sistemi di calcolo. Conseguentemente, viene prevalentemente utilizzato se si è interessati al moto di un fluido e non alla sua qualità grafica. Nel panorama attuale, sono presenti altri metodi facenti parte di uno dei punti di vista sopra citati e trattati nel capitolo 1, tuttavia, i tre proposti risultano essere i più diffusi. Ricerche future si potrebbero sviluppare su tale campo così come l'implementazione e la creazione di un proprio simulatore di fluidi creato a partire dagli elementi esposti in questa tesi.

Bibliografia

[BB08] C. Batty e R. Bridson, “Accurate viscous free surfaces for buckling, coiling, and rotating liquids.” in *Proc. ACM. SIGGRAPH/Eurographics Symp. Comp. Anim.,* SCA ’08, pp. 219-228, 2008.

[BBB07] C. Batty, F. Bertails e R. Bridson, “A fast variational framework for accurate solid-fluid coupling”, *ACM Trans. Graph. (Proc. SIGGRAPH),* 26(3), 2007.

[BLE16] Blender Foundation, Blender Reference Manual, https://www.blender.org/manual/contents.html

[BR86] J. U. Brackbill e H. M. Ruppel, “FLIP: a method for adaptively zoned, particle-in-cell calculations of fluids flow in two dimensions. “, *J. Comp. Phys.*, 65:314-343, 1986.

[BRI03] R. Bridson, *Computational Aspects of Dynamic Surfaces,* PhD thesis, Stanford University, June 2003.

[BRI15] R. Bridson, *Fluid Simulation for Computer Graphics,* CRC Press, Taylor & Francis Group, Boca Raton, FL, 2015.

[CMF12] N. Chentanez e M. Müller-Fisher, “A multigrid fluid pressure solver handling separating solid boundary conditions”, *IEEE Trans. Vis. Comp. Graph.*, 18(8):1911-1201, 2012.

[CMIT02] M. Carlson, P. Mucha, R. Van Horn III e G. Turk, “Melting and flowing”, in *Proc. ACM SIGGRAPH/Eurographics Symp. Comp. Anim.,* pp. 167-174, 2002.

[EB15] E. Edward e R. Bridson, “The discretely-discontinuous Galerkin coarse grid for domain decomposition”, *Tchnical report,* arXiv: 1504.00907, 2015.

[FF01] N. Foster e R. Fedkiw, “Practical animation of liquids”, in *Proc. SIGGRAPH,* pp. 23-30, 2001.

[FR86] A. Fournier e W. T. Reeves. “A simple model of ocean waves”, In *Proc. SIGGRAPH*, pp. 75-84, 1986.

[FSJ01] R. Fedkiw, J. Stam e H. W. Jensen, “Visual simulation of smoke”, in *Proc. SIGGRAPH,* pp. 15-22, 2001.

[GSLF05] E. Guendelman, A. Selle, F. Losasso e R. Fedkiw, “Coupling water and smoke to thin deformable and rigid cells“, *ACM Trans. Graph.* (*SIGGRAPH Proc.*), 24(3):973-981, 2005.

[HAR63] F. H. Harlow, “The particle-in-cell method for numerical solution of problems in fluid dynamics.” in *Experimental arithmetic, high-speed computations and mathematics*, 1963.

[HW65] F. Harlow e J. Welch, “Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface”. *Phys. Fluids*, 8:2182-2189, 1965.

[IRG08] F. Irgens, *Continuum Mechanics*, Springer, 2008.

[JBS06] M. Jones, A. Bærentzen e M. Sramek, “3D distance fields: A survey of techniques and applications“, *IEEE Trans. Vis. Comp. Graphics*, 2006.

[MON92] J. J. Monaghan, “Smoothed Particle Hydrodynamics”, *Annu. Rev. Astron. Astrophys.”,* 30:543-74, 1992.

[MPC16] MPC Film, “Poseidon: set extensions and CG environment”, http://www.moving-picture.com/film/filmography/poseidon, 2016

[MUS11] K. Museth, “Db grid: a novel dynamic blocked grid for sparse high-resolution volumes and level sets”, in *ACM SIGGRAPH 2011 Talks, SIGGRAPH* 11, pp. 51:1-51:1, 2011.7

[MUS13] K. Museth, “Vdb: High-resolution sparse volumes with dynamic topology”, *ACM Trans. Graph.*, 32(3):27:1-27:22, 2013.

[OF02] S. Osher e R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Springer-Verlag, New York, NY, 2002.

[RAL62] A. Ralston, “Runge-Kutta methods with minimum error bounds.” *Mathematics of computation*, 16(80):431-437, 1962.

[RNGF03] N. Rasmussen, D. Nguyen, W. Geiger e R. Fedkiw, “Smoke simulation for large scale phenomena”, *ACM Trans. Graph. (Proc. SIGGRAPH)*, 22:703-707, 2003.

[STA99] J. Stam, “Stable fluids”, in *Proc. SIGGRAPH*, pp. 121-128, 1999.

[SU94] J. Steinhoff e D. Underhill, “Modification of the Euler Equations for Vorticity Confinement: Application to the Computation of Interacting Vortex Rings”, *Phys. of Fluids*, 6(8):2738-2744, 1994.

[TES04] J. Tessendorf. “Simulating ocean water.”, *SIGGRAPH Course Notes*,1999-2004.

[TSA02] Y.-H. R. Tsai, “Rapid and accurate computation of the distance function using grids”, in *J. Comput. Phys*, 178(1):175-195, 2002.

[YJ11] Yuanxun Bill Bao e Justin Meskas, “Lattice Boltzmann Method for Fluid Simulations“, 2011.