

Politechnika Wrocławска
Wydział Informatyki i Zarządzania

Praca inżynierska

Symulacja komputerowa mechaniki płynów

FLUID MECHANICS COMPUTER SIMULATION

Jakub Leszczak

Opiekun pracy: Dr inż. Maciej Drwal

Wrocław, grudzień 2015

Streszczenie

Praca ta stanowi wprowadzenie do tematyki symulacji komputerowej mechaniki płynów. Opisane są równania Naviera-Stokesa, sposoby ich dyskretyzacji oraz algorytmy numeryczne, bazujące na tych równaniach. Praca zawiera także opis eksperymentów przeprowadzonych w celu zbadania działania takich algorytmów. W podsumowaniu zawarte zostały wnioski z eksperymentów oraz przykładowe możliwości dalszego rozwoju pracy nad simulacjami.

This work is an introduction to the field of computer fluid simulations. It describes Navier-Stokes' equations, methods of their discretization and algorithms of fluid simulation derived from these equations. This work also contains a report on series of experiments conducted in order to analyse the properties of such algorithms. The last part of the thesis contains conclusions from experiments and proposed directions of future work and development in the field of fluid simulations.

Słowa Kluczowe: algorytmy numeryczne, równania Naviera-Stokesa, symulacja procesów fizycznych, mechanika płynów, grafika komputerowa

Spis treści

1 Wprowadzenie	1
1.1 Definicja symulacji komputerowej	1
1.2 Zastosowania symulacji mechaniki plynów	1
1.3 Równania Naviera-Stokesa	2
1.3.1 Symbole	2
1.3.2 Równanie ruchu	3
1.3.3 Warunek nieściśliwości	5
1.4 Warunki brzegowe	7
1.5 Środowisko implementacji	7
1.6 Uwagi bibliograficzne	8
2 Algorytmy symulacji	11
2.1 Dyskretyzacja	11
2.2 Siatka Marker and Cell	13
2.3 Algorytm adwekcji	15
2.3.1 Wielkość kroku Δt	16
2.3.2 Dyssypacja	17
2.3.3 Ekstrapolacja	17
2.4 Algorytm projekcji	18

SPIS TREŚCI

2.4.1	Zebranie w postać macierzową	20
2.4.2	Nazwa „projekcja”	21
2.5	Reprezentacja płynu	21
2.6	Schemat symulacji	23
3	Eksperymenty	25
3.1	Woda stojąca	25
3.1.1	Wydajność	25
3.1.2	Zbieżność metod rozwiązywania układów równań	26
3.1.3	Poprawność obliczeń	28
3.2	Falowanie wody	29
3.2.1	Wizualizacja	29
3.3	Ruch wirowy	30
3.3.1	Wizualizacja	31
3.3.2	Dyssypacja	32
3.3.3	Ciśnienie	33
3.4	Zwężenie	35
3.4.1	Wizualizacja	35
3.4.2	Nieściśliwość wody	36
3.5	Woda leżąca się z kranu	36
3.5.1	Wizualizacja	37

SPIS TREŚCI

3.5.2 Problem zatrzymania wody	37
3.5.3 Problem braku oporu	38
3.6 Wnioski	39
4 Podsumowanie	41
4.1 Reprezentacja płynu	41
4.2 Algorytm adwekcji	41
4.3 Algorytm projekcji	42
4.4 Mieszanie płynow	42
Bibliografia	43
Spis rysunków	45
Skorowidz	47

SPIS TREŚCI

Rozdział 1

Wprowadzenie

1.1 Definicja symulacji komputerowej

Symulacja komputerowa polega na wykorzystaniu modelu matematycznego do zbadania interesującego nas zjawiska. Symulacja odbywa się poprzez przeprowadzenie obliczeń przy pomocy odpowiedniego programu komputerowego. Techniki symulacyjne są szczególnie przydatne tam, gdzie analityczne wyznaczenie rozwiązania byłoby zbyt pracochłonne, a niekiedy nawet niemożliwe. Symulacji mogą podlegać różne zjawiska fizyczne, dla których znane są odpowiednie modele. Takim zjawiskiem, którego dotyczy niniejsza praca, jest ruch płynu.

1.2 Zastosowania symulacji mechaniki płynów

Symulacje mechaniki płynów znajdują zastosowanie w wielu różnych dziedzinach. Wykorzystuje się je w grafice komputerowej, w grach, w animacjach oraz w pewnym stopniu w przemyśle. Metody numeryczne przedstawione w tej pracy oparte są na równaniach Naviera-Stokesa. Są to równania różniczkowe, których rozwiązania analityczne nie są znane, dlatego też zmuszeni jesteśmy do ich dyskretyzacji i przybliżania. W związku z tym często aby uzyskać satysfakcjonującą nas precyzję musimy uzbroić się w cierpliwość, bo wymagana jest duża moc obliczeniowa oraz dużo czasu.

Dlatego też w grach komputerowych, gdzie wymagana jest symulacja w czasie rzeczywistym, stosowane są uproszczone modele matematyczne, niekoniecznie będące w zgodzie z prawami fizyki, np. aproksymację mapy wysokości (ang. Heightfield Approximation [4]).

Natomiast w zastosowaniach przemysłowych bardzo często dużo prostszym i tańszym sposobem przeprowadzania symulacji jest ich realizacja fizyczna. Przykładowo, dużo łatwiej stworzyć jest tunel aerodynamiczny, umieścić w nim interesujący nas prototyp samochodu i zbadać ruchy powietrza w taki sposób, niż tworząc symulację komputerową.

Symulacje komputerowe mechaniki płynów są obiektem wielu prac naukowych. Pożądany jest by symulacje te były możliwie jak najszybsze i jak najdokładniejsze, a wraz z rosnącą siłą obliczeniową sprzętów komputerowych zastosowania takich symulacji w praktyce będą coraz większe.

1.3 Równania Naviera-Stokesa

Podstawą symulacji mechaniki płynów w niniejszej pracy są nieściśliwe równania Naviera-Stokesa. Mają one postać:

$$\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \nabla \vec{u}) + \nu \nabla^2 \vec{u} + \vec{g} - \frac{1}{\rho} \nabla p \quad (1.1)$$

$$\nabla \cdot \vec{u} = 0 \quad (1.2)$$

1.3.1 Symbole

Oznaczeniem \vec{u} tradycyjnie w mechanice płynów oznaczona jest prędkość. Oznaczenie to ma taką przewagę nad popularnym w innych działach fizyki \vec{v} , że tworzy całkiem użyteczną konwencję, w której wektor trójwymiarowy prędkości oznaczony jest (u, v, w) , zamiast (v_x, v_y, v_z) , co skracia zapis i sprawia, że równania są czytelniejsze.

Symbol ν oznacza lepkość kinematyczną. Jest to miara lepkości płynu – oporu płynu na deformacje. Dla przykładu, miód ma dużą lepkość, a alkohol niską.

Symbol \vec{g} standardowo oznacza grawitację. W ogólnym przypadku, tak jak i w tych równaniach, jest to jednak oznaczenie wszelkich sił zewnętrznych działających na płyn.

Symbol ρ oznacza gęstość płynu. Dla wody jest to około 1000 kg/m^3 . Symbol p oznacza ciśnienie. Co ciekawe, w równaniach Naviera-Stokesa można zauważać, że przy gradiencie ciśnienia widnieje znak minus. W równaniach tych ciśnienie nie jest traktowane jako siła działająca na powierzchnię cieczy, ale jako siła, która utrzymuje ciecz nieściśliwą (sprawia, że płyn nie zmienia swojej objętości). Jest to więc siła przeciwnie skierowana do siły działającej na powierzchnię i, zgodnie z trzecią zasadą dynamiki Newtona, ma taką samą wartość.

1.3.2 Równanie ruchu

Pierwsze równanie Naviera-Stokesa (1.1) jest nazywane **równaniem ruchu**. Tak naprawdę jest to dobrze znane równanie Newtona $\vec{F} = m\vec{a}$. Mówiąc ono nam jak płyn przyspiesza zależnie od działających na niego sił. Spróbujmy więc dowiedzieć się skąd to równanie ruchu przybrało taką postać.

Płyn możemy sobie wyobrazić jako zbiór cząstek. Każda cząstka posiada masę m , objętość V i prędkość \vec{u} . Aby przemieścić ten zbiór w czasie musimy wyliczyć siły działające na każdą cząstkę. Następnie wyliczyć ich przyspieszenie, a dzięki temu będziemy znali ich ruch.

Przyspieszenie \vec{a} to nic innego jak pochodna prędkości po czasie: $\vec{a} = d\vec{u}/dt$. Równanie ruchu cząstki przyjmuje więc postać:

$$m \frac{d\vec{u}}{dt} = \vec{F} \quad (1.3)$$

Pytanie jakie pozostaje to jakie siły działają na cząstkę i jak je wyliczyć. Pierwszą siłą jest grawitacja: $m\vec{g}$ (i jak wspomniane było wcześniej, także wszelkie inne siły zewnętrzne).

Źródłem drugiej siły działającej na cząstkę jest ciśnienie. Region o wyższym ciśnieniu naciska na region o niższym ciśnieniu. Tak naprawdę tym co nas interesuje nie jest samo ciśnienie, ale różnica ciśnień między regionami (gradient ciśnienia): $-\nabla p$. Aby uzyskać

się należy gradient ten scałkować na całej objętości cząstki. Jako proste przybliżenie przemnożymy gradient przez objętość V .

Trzecią siłą działającą na płyn jest lepkość. Lepki płyn próbuje stawiać opór wszelkim deformacjom. Jest to siła która stara się by każda cząstka poruszała się ze średnią prędkością jej sąsiadów – stara się minimalizować różnice między prędkościami sąsiadujących cząstek. Operatorem różniczkującym który mierzy różnicę między ośrodkiem a średnią wokół jest laplasjan ∇^2 . Podobnie jak w przypadku ciśnienia, laplasjan lepkości należy scałkować po całej objętości cząstki – przybliżamy to mnożąc przez objętość. Dodatkowo mnożymy to wszystko przez lepkość dynamiczną cieczy μ (jest to współczynnik który wyraża jak bardzo lepki jest płyn).

Zbierając wszystkie te siły w jedno, oto równanie które otrzymujemy:

$$m \frac{d\vec{u}}{dt} = m\vec{g} - V\nabla p + V\mu\nabla^2\vec{u} \quad (1.4)$$

Dzieląc przez objętość V obie strony równania i pamiętając że $\rho = m/V$:

$$\rho \frac{d\vec{u}}{dt} = \rho\vec{g} - \nabla p + \mu\nabla^2\vec{u} \quad (1.5)$$

Następnie dzieląc przez gęstość ρ :

$$\frac{d\vec{u}}{dt} = \vec{g} - \frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2\vec{u} \quad (1.6)$$

Podstawiając wzór na zależność lepkości kinematycznej od dynamicznej $\nu = \mu/\rho$:

$$\frac{d\vec{u}}{dt} = \vec{g} - \frac{1}{\rho}\nabla p + \nu\nabla^2\vec{u} \quad (1.7)$$

Już prawie otrzymaliśmy upragniony wynik. Ostatnią rzeczą którą musimy zrobić jest zamiana pochodnej zupełnej na pochodne cząstkowe. Zgodnie ze wzorem zależności między pochodną zupełną a cząstkową:

$$\frac{d}{dt}f(\vec{x}) = \frac{\partial f}{\partial x_1} \frac{dx_1}{dt} + \frac{\partial f}{\partial x_2} \frac{dx_2}{dt} + \dots + \frac{\partial f}{\partial x_n} \frac{dx_n}{dt} \quad (1.8)$$

Możemy obliczyć, przyjmując oznaczenie $\vec{u} = (u, v, w)$:

$$\frac{d\vec{u}}{dt} = \frac{\partial \vec{u}}{\partial t} \frac{dt}{dt} + \frac{\partial \vec{u}}{\partial x} \frac{dx}{dt} + \frac{\partial \vec{u}}{\partial y} \frac{dy}{dt} + \frac{\partial \vec{u}}{\partial z} \frac{dz}{dt} \quad (1.9)$$

Korzystając z podstawowego wzoru fizycznego mówiącego, że prędkość jest pochodną przemieszczenia po czasie $v = \frac{dx}{dt}$ otrzymujemy:

$$\frac{d\vec{u}}{dt} = \frac{\partial \vec{u}}{\partial t} + \frac{\partial \vec{u}}{\partial x}u + \frac{\partial \vec{u}}{\partial y}v + \frac{\partial \vec{u}}{\partial z}w \quad (1.10)$$

I dalej przekształcając:

$$\frac{d\vec{u}}{dt} = \frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} \quad (1.11)$$

Podstawiając równanie (1.11) do równania (1.7) i przestawiając wyrazy otrzymujemy wynik końcowy będący równaniem ruchu płynów.

$$\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \nabla \vec{u}) + \nu \nabla^2 \vec{u} + \vec{g} - \frac{1}{\rho} \nabla p \quad (1.12)$$

1.3.3 Warunek nieściśliwości

Drugie równanie Naviera-Stokesa (1.2) to tak zwany **warunek nieściśliwości**. Spełnienie tego równania przez wszystkie nasze cząstki w płynie zagwarantuje nam, że nie zmienią on swojej objętości. W rzeczywistości wszystkie płyny zmieniają swoją objętość pod wpływem sił zewnętrznych, temperatury oraz z powodu rozchodzących się fal dźwiękowych. Nie mniej jednak, płyny nie zmieniają swojej objętości bardzo mocno. Fizycznie jest to niemal niemożliwe by znacząco zmienić objętość wody. Nawet powietrze nie zmienia swojej objętości znacząco, dopóki nie ma się do czynienia z ogromnymi siłami rzędu wybuchu bomby. W praktyce symulacja płynów które są ściśliwe jest bardzo skomplikowana i kosztowna obliczeniowo, a dodatkowo zapotrzebowanie na takie symulacje w codziennym życiu jest znikome.

Dlatego też w symulacjach komputerowych wszystkie płyny są traktowane jako nieściśliwe. W jaki sposób można wyrazić to matematycznie? Wybierzmy region płynu, jego objętość oznaczmy przez Ω , a jego powierzchnię przez $\partial\Omega$. Możemy zmierzyć, jak szybko objętość tego regionu się zmienia przez całkowanie składowej normalnej prędkości po

powierzchni:

$$\frac{d}{dt} Volume(\Omega) = \iint_{\partial\Omega} \vec{u} \cdot \hat{n} \quad (1.13)$$

gdzie \hat{n} to wektor jednostkowy normalnej składowej prędkości. Aby płyn był nieściśliwy zmiana objętości musi wynosić zero:

$$\iint_{\partial\Omega} \vec{u} \cdot \hat{n} = 0 \quad (1.14)$$

Teraz możemy skorzystać z twierdzenia Ostrogradskiego-Gaussa i zamienić to na całkę po objętości:

$$\iiint_{\Omega} \nabla \cdot \vec{u} = 0 \quad (1.15)$$

Ostatecznie wiemy, że równanie to musi być spełnione dla dowolnego regionu płynu Ω . Jedyna funkcja, która dla dowolnej objętości daje wynik zero, to właśnie zero. Dlatego też musi być spełnione równanie:

$$\nabla \cdot \vec{u} = 0 \quad (1.16)$$

W ten sposób uzyskaliśmy równanie warunku nieściśliwości – drugie z równań Naviera-Stokesa. Pole wektorowe, które spełnia warunek nieściśliwości nazywa się z angielskiego **divergence-free**. Brak jest polskiego odpowiednika tej nazwy. Można się pokusić o określenie bezdywergencyjny albo pozbawiony rozbieżności.

Jednym z głównych zadań symulacji jest zapewnienie aby pole wektorowe było bezdywergencyjne. Tutaj właśnie z pomocą przychodzi nam ciśnienie. Ciśnienie to siła, która za wszelką cenę utrzymuje nasze pole wektorowe bezdywergencyjne. W jaki więc sposób uzależnić ciśnienie, które pojawia się jedynie w równaniu ruchu, od warunku nieściśliwości? Obliczmy dywergencję równania ruchu:

$$\nabla \cdot \frac{\partial \vec{u}}{\partial t} = -\nabla \cdot (\vec{u} \cdot \nabla \vec{u}) + \nabla \cdot (\nu \nabla^2 \vec{u} + \vec{g}) - \nabla \cdot \frac{1}{\rho} \nabla p \quad (1.17)$$

Mogemy zamienić kolejnością pochodną cząstkową z gradientem, otrzymując (wyrażenie z lewej strony równania):

$$\frac{\partial}{\partial t} \nabla \cdot \vec{u} \quad (1.18)$$

Z warunku nieściśliwości (1.2) wiemy, że gradient ten wynosi zero, tak więc wstawiając zero do równania (1.17) i przestawiając wyrazy otrzymujemy:

$$\nabla \cdot \frac{1}{\rho} \nabla p = \nabla \cdot (-\vec{u} \cdot \nabla \vec{u} + \nabla \cdot \nu \nabla^2 \vec{u} + \vec{g}) \quad (1.19)$$

Do dokładnego zastosowania równania na obliczanie ciśnienia powrócimy w dalszej części tej pracy w rozdziale poświęconym projekcji.

1.4 Warunki brzegowe

Do tej pory opisaliśmy co dzieje się wewnątrz płynu. Jednak większość problemów związanych z symulowaniem płynów polega na poprawnym opisaniu warunków brzegowych. Skupimy się tutaj na dwóch przypadkach: nieruchomych ścianach i wolnej przestrzeni.

Przypadek brzegowy z nieruchomą ścianą to przypadek, gdy płyn jest w bezpośrednim kontakcie ze ścianą. W związku z tym normalna składowa prędkości w tym miejscu musi wynosić zero:

$$\vec{u} \cdot \hat{n} = 0 \quad (1.20)$$

\hat{n} jest normalną do powierzchni ściany. Równanie to ma bezpośredni wpływ na ciśnienie p .

Drugim przypadkiem brzegowym jest wolna przestrzeń. Wolna przestrzeń znajduje się wszędzie tam, gdzie nie znajduje się płyn, ani ciało stałe (ściana). Jako że symulacja powietrza w wolnej przestrzeni nie jest konieczna dla naszych celów, to naturalnie możemy przyjąć, że ciśnienie w tych miejscach równe jest zero $p = 0$. Dokładny opis obliczania ciśnienia i obsługi warunków brzegowych pojawi się w rozdziale poświęconym projekcji 2.4.

1.5 Środowisko implementacji

W celu implementacji symulacji mechaniki płynów użyte zostały technologie takie jak: C++11, OpenGL 3.3, GLFW, GLM oraz freetype. Trzy ostatnie to biblioteki języka

programowania C++. GLFW pozwala na tworzenie prostego okna interfejsu graficznego oraz na przechwytywanie zdarzeń (takich jak ruch myszą, naciśnięcie klawiszy klawiatury). GLM to biblioteka implementująca operacje numeryczne na macierzach i wektorach. Freetype pozwala na łatwą obsługę czcionek i renderowanie tekstu.

W aplikacji mającej na celu symulowanie mechaniki płynów, w szczególności zależeć powinno nam na wydajności, dlatego też zdecydowanym faworytem jest język C++ w wersji 11 [6]. Jest to język wydajny, niskopoziomowy i dający duże możliwości optymalizacji. Wersja 11 dodatkowo wprowadza wiele użytecznych funkcjonalności, które ułatwiają programowanie w porównaniu do starszych wersji tego języka. Dodatkowo użyta została biblioteka OpenGL do renderowania grafiki, ponieważ jest ona multiplatformowa, szybka i daje dużą kontrolę programistie nad tym w jaki sposób renderowanie powinno się odbywać.

1.6 Uwagi bibliograficzne

Główną pracą, na której wzorowana była ta praca inżynierska były notatki „*Fluid Simulation*” [4]. W pracy tej dokładnie opisane są podstawy związane z symulacją komputerową mechaniki płynów: opis równań, metody numeryczne, algorytmy i sposoby reprezentacji płynów.

Następną pracą jest książka „*Numerical Simulation in Fluid Dynamics*” [12]. Praca ta także dokładnie opisuje podstawy, a dodatkowo zawiera pseudokody wielu algorytmów oraz rysunki co może pomóc w zrozumieniu ich działania. Do tego opisana jest tam metoda reprezentacji płynu metodą markerów, sposoby zrównoleglania algorytmów oraz transport cieplny.

Reprezentacja płynu metodą markerów jest metodą dosyć starą, a nowszą jest reprezentacja płynu przy użyciu **level sets** opisanych we wcześniej wspomnianych notatkach [4] oraz dokładniej w pracach „*Level Sets Methods and Fast Marching Methods*” [16] i „*Level*

Set Methods for Fluid Interfaces” [17].

Ciekawą pozycją w bibliografii jest także praca dotycząca sposobu ulepszenia algorytmu adwekcji „*Advections with Significantly Reduced Dissipation and Diffusion*” [5] by zmniejszyć dyssypację. Kwestie dotyczące ulepszenia algorytmu projekcji poruszane są także w pracach: „*A Second-Order Projection Method for Variable-Density Flows*” [3], „*A High-Order Projection Method for Tracking Fluid Interfaces in Variable Density Incompressible Flows*” [7].

Pozostałymi pracami pełnymi interesujących informacji, które były niezmiernie pomocne przy tworzeniu tej pracy inżynierskiej były: „*Nvidia GPU Gems*” [10], „*Real-Time Fluid Dynamics for Games*” [18], „*Fluid Simulation Tutorial*” [14]. Bardzo pomocne były także kursy OpenGL'a: „*Nvidia GPU Gems*” [1], „*Real-Time Fluid Dynamics for Games*” autorstwa Stama [2], „*Fluid Simulation Tutorial*” [15].

Rozdział 2

Algorytmy symulacji

W poprzednim rozdziale opisane zostały równania Naviera-Stokesa. Aby możliwe było ich wykorzystanie w symulacji komputerowej, należy je zdyskretyzować. W tym celu przeanalizujemy kolejne składniki równania ruchu (1.1).

2.1 Dyskretyzacja

Aby zdyskretyzować równanie ruchu Naviera-Stokesa (1.1) możemy posłużyć się metodą dyskretyzacji Eulera. Otrzymujemy w ten sposób równanie:

$$\vec{u}_{n+1} = \vec{u}_n + [-(\vec{u} \cdot \nabla \vec{u}) + \nu \nabla^2 \vec{u} + \vec{g} - \frac{1}{\rho} \nabla p] \Delta t \quad (2.1)$$

Możemy to równanie rozdzielić na kilka prostszych, by ostatecznie je ze sobą zsumować:

$$\vec{u}_{n+1} = \vec{u}_n - (\vec{u} \cdot \nabla \vec{u}) \Delta t \quad (2.2)$$

$$\vec{u}_{n+1} = \vec{u}_n + (\nu \nabla^2 \vec{u}) \Delta t \quad (2.3)$$

$$\vec{u}_{n+1} = \vec{u}_n + \vec{g} \Delta t \quad (2.4)$$

$$\vec{u}_{n+1} = \vec{u}_n - \frac{1}{\rho} \nabla p \Delta t \quad (2.5)$$

W tym miejscu pojawia się pierwsza niespodziewana operacja. Zamiast wykonywać te wszystkie operacje równolegle, możemy je uszeregować jedna po drugiej (przy Δt dążącym do zera jest to matematycznie poprawne) uzyskując:

$$\vec{u}_{n+1} = \vec{u}_n - (\vec{u} \cdot \nabla \vec{u}) \Delta t \quad (2.6)$$

$$\vec{u}_{n+2} = \vec{u}_{n+1} + (\nu \nabla^2 \vec{u}) \Delta t \quad (2.7)$$

$$\vec{u}_{n+3} = \vec{u}_{n+2} + \vec{g} \Delta t \quad (2.8)$$

$$\vec{u}_{n+4} = \vec{u}_{n+3} - \frac{1}{\rho} \nabla p \Delta t \quad (2.9)$$

Pojawia się pytanie: jaki jest cel takiego zabiegu i czy uzyskanie dokładniejszego wyniku byłoby możliwe przy użyciu poprzedniej postaci równań. Odpowiedź jest taka, że ostatecznie do rozwiązywania tych równań posłużymy się lepszymi algorytmami aniżeli dyskretyzacją Eulera. Z natury tych algorytmów wynika, że aby poprawnie funkcjonowały, należy zachować odpowiednią kolejność obliczeń; nie możemy ich użyć równolegle. Dokładniejsze wyjaśnienie przyczyny będzie opisane w sekcji o algorytmie projekcji 2.4.

W ten oto sposób uzyskaliśmy cztery równania i każde z nich będziemy rozwiązywać osobnym algorytmem. Algorytmy te realizują kolejno następujące operacje: adwekcję, dyfuzję, działanie sił zewnętrznych i projekcję prędkości. W dalszej części pracy zostaną wyjaśnione znaczenia nazw tych operacji. Pozostaje jeszcze pytanie, co z drugim równaniem Naviera-Stokesa (1.2) oraz warunkami brzegowymi omówionymi w podrozdziale 1.4. Tak jak już wcześniej było wspomniane, warunek nieściśliwości oraz warunki brzegowe mają bezpośredni wpływ na ciśnienie, tak więc równanie to będzie wykorzystane w algorytmie projekcji.

Podsumowując, ostatecznie schemat symulacji wygląda następująco:

1. Ustaw warunki początkowe: pole \vec{u}_0
2. Wybierz odpowiedni krok Δt
3. Wykonaj algorytm adwekcji: $\vec{u}_{n+1} = A(u_n, \Delta t)$
4. Wykonaj algorytm dyfuzji: $\vec{u}_{n+2} = D(u_{n+1}, \Delta t)$
5. Wykonaj algorytm sił zewnętrznych: $\vec{u}_{n+3} = G(u_{n+2}, \Delta t)$
6. Wykonaj algorytm projekcji: $\vec{u}_{n+4} = P(u_{n+3}, \Delta t)$

Powtarzaj kroki 2 – 6.

Algorytm sił zewnętrznych wymaga jedynie wykonania prostej dyskretyzacji Eulera, gdyż \vec{g} to wartość stała – czyli wystarczy dodać do pola prędkości $\vec{g}\Delta t$. Pozostałe algorytmy są jednak niestabilne, jeśli zostaną wykonane metodą dyskretyzacji Eulera [4], dlatego stosuje się inne – stabilne – algorytmy, opisane w dalszej części tego rozdziału.

2.2 Siatka Marker and Cell

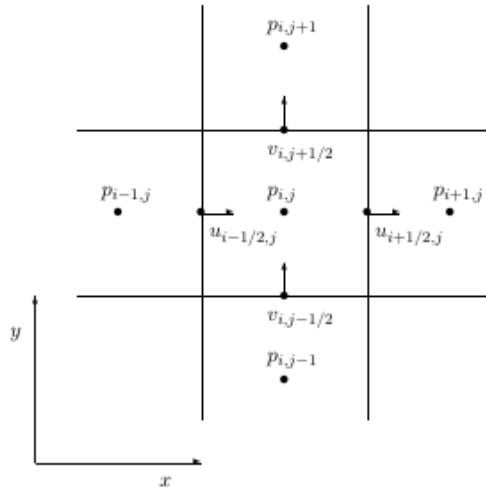
Kolejną rzeczą, którą musimy zrobić, jeszcze przed dokładnym opisaniem algorytmów symulacji, jest opisanie, w jaki sposób będziemy przechowywać nasze dane odnośnie pola wektorowego prędkości oraz pól skalarnych ciśnienia, gęstości, temperatury itp.

Podejścia są dwa: podejście Lagrange'a i podejście Eulera. Pierwsze z nich polega na prostym i intuicyjnym reprezentowaniu każdej cząstki z osobna i przemieszczaniu ich w czasie. Drugie podejście z kolei działa odwrotnie. Zamiast śledzić ruch cząstek, będziemy śledzić zmiany prędkości i innych wartości, zawsze w tych samych miejscach. Czyli zamiast badać, gdzie dana cząstka znajdzie się w danym momencie, badamy jakie cząstki znajdują się w danym miejscu w danym momencie. Dla celów tej pracy wykorzystane zostało podejście Eulera, które jest też podejściem popularniejszym w symulacjach mechaniki płynów.

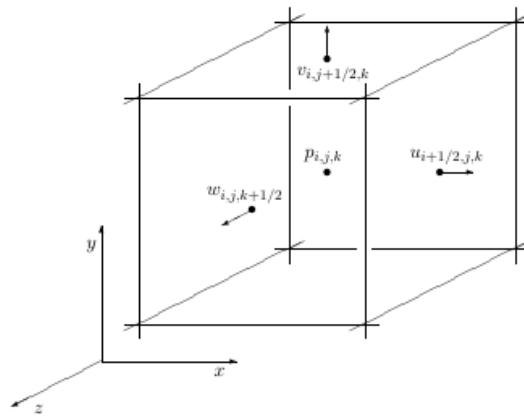
W celu reprezentacji naszego płynu posłużymy się siatką Marker and Cell, opisaną po raz pierwszy w pracy „Numerical Calculation of Time-Dependent Viscous Incompressible

Flow of Fluid with Free Surface” [9].

Siatka ta nazywana **MAC gridem**, dzieli symulowany obszar na sześcianny (kwadraty w 2D) zwane komórkami. Wartości skalarne są zapisywane w centrach tych sześciianów, a wartości wektorowe prędkości na środkach ścian sześciianów. Może nie wydawać się to bardzo istotne, ale w rzeczywistości pomysł ten ma kluczowe znaczenie dla efektywnej symulacji. Przechowywanie prędkości na ścianach, zamiast w centrach sześciianów, pozwala na znacznie dokładniejsze zdyskretyzowanie równań ciśnienia w algorytmie projekcji.



Rysunek 2.1: Siatka MAC grid w 2D [4]



Rysunek 2.2: Komórka siatki MAC grid w 3D [4]

2.3 Algorytm adwekcji

Pierwszym niezbędnym algorytmem do symulacji jest algorytm adwekcji. Odpowiednie równanie miało postać:

$$\vec{u}_{n+1} = \vec{u}_n - (\vec{u} \cdot \nabla \vec{u}) \Delta t \quad (2.10)$$

Bardzo podobne równanie, które tak naprawdę już wcześniej się pojawiło (1.11) w postaci niezdyskretyzowanej, mówi nam, że powyższe równanie jest równoważne następującemu:

$$\frac{d\vec{u}}{dt} = 0 \quad (2.11)$$

To równanie oznacza, że zmiana prędkości w czasie wynosi zero. Czyli, zgodnie z podstawowymi zasadami dynamiki Newtona, w tym algorytmie nie działają na płyn żadne siły zewnętrzne. Algorytm adwekcji polega więc na przemieszczaniu płynu do przodu po polu prędkości.

Gdybyśmy stosowali podejście Lagrange'a algorytm ten byłby po prostu przemieszczeniem częstek o dystans $\Delta \vec{x} = \vec{u} \Delta t$. W przypadku podejścia Eulera i siatki MAC grid algorytm ten lekko się komplikuje. Jeśli próbowałibyśmy przemieścić centrum każdej komórki o dystans $\Delta \vec{x}$, to prawie nigdy nie trafiemy w centrum innej komórki, co sprawiłoby, że w żaden sposób niemożliwe byłoby zapisanie takiej informacji. Dlatego też stosuje się pewien trik, który bazuje na spostrzeżeniu, że pole prędkości w małym otoczeniu jest prawie jednorodne. Dlatego też, cofając się o dystans $\Delta \vec{x}$ wstecz, powinniśmy trafić na niemal identyczną prędkość z jakiej wystartowaliśmy (szczególnie gdy Δt dąży do zera).

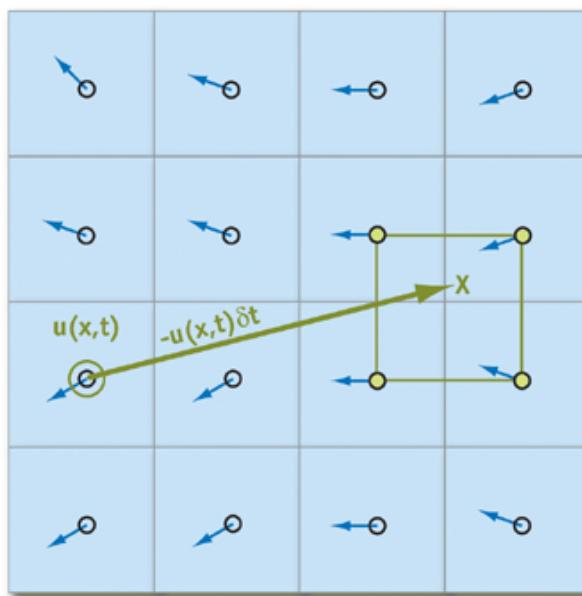
Adwekcja do przodu:

$$\vec{u}(\vec{x} + \Delta \vec{x}, t + \Delta t) = \vec{u}(\vec{x}, t) \quad (2.12)$$

Adwekcja wstecz:

$$\vec{u}(\vec{x}, t + \Delta t) = \vec{u}(\vec{x} - \Delta \vec{x}, t) \quad (2.13)$$

W związku z tym, w praktyce stosuje się adwekcję wstecz. Czyli zamiast przemieszczać centrum komórki do przodu, wykonujemy przemieszczenie z centrum komórki wstecz i zakładamy, że wartość skalarna (lub wektorowa, w przypadku pola prędkości) jaką tam zastąpiemy jest mniej więcej tą wartością, która trafiłaby do centrum tej komórki, gdybyśmy wykonali adwekcję do przodu. W ten sposób także prawie nigdy nie trafimy do centrum innej komórki, ale pomiędzy kilkoma komórkami MAC grida. Dlatego też by uzyskać wartość, jaka znajduje się w tym punkcie, dokonuje się **interpolacji** (liczymy średnią ważoną ze wszystkich wartości w centrach sąsiadujących komórek).



Rysunek 2.3: Algorytm adwekcji [10]

2.3.1 Wielkość kroku Δt

Chcielibyśmy oczywiście, aby wyniki naszych symulacji były jak najdokładniejsze, ale też żeby obliczenia wykonywały się w rozsądny czasie. Dlatego też wymagane jest znalezienie złotego środka – takiej wartości Δt , która stanowi kompromis pomiędzy dokładnością a czasem obliczeń. Ponieważ wykonujemy adwekcję wstecz, w której wyszliśmy z założenia, że jest ona dokładna dopóki przemieszczamy się po najbliższym otoczeniu

centrum komórki, najlepiej by krok Δx był możliwie jak najmniejszy. W praktyce zastosowany został taki krok Δx , aby nie przekroczyć szerokości jednej komórki. Z tego wynika, że Δt jest mniejsza niż $\Delta x / \max(\vec{u})$.

2.3.2 Dyssypacja

Wcześniej zostało opisane, że podczas adwekcji niezbędne jest wykonywanie interpolacji (uśredniania). Niesie to ze sobą pewną nieuniknioną przypadłość, nazywaną dyssypacją. Dyssypacja jest to rozpraszanie się różnic. W praktyce dyssypacja powoduje, że różnice prędkości są zmniejszane, aż w końcu cały płyn będzie jednorodny. Jest to rzecz często niepożądana i nieunikniona (choć istnieją metody zmniejszające wpływ dyssypacji na symulację [5]), nie mniej jednak, w niektórych sytuacjach, ku zaskoczeniu, okazuje się pomocna. Przykład wykorzystania dyssypacji na naszą korzyść jest pokazany w eksperymencie 3.5.3.

Dokładnie problem dyssypacji jest opisany w [4]. Jest w nich wspomniane, że dyssypacja jest wprost proporcjonalna do kwadratu kroku czasowego Δt^2 . Zachowuje się ona więc dokładnie jak lepkość.

Ponieważ więc algorytm adwekcji, mimo wszystko, symuluje efekt dyfuzji, a nie jest nam potrzebne dokładne odzwierciedlenie lepkości (gdyż w pracy tej skupiamy się na symulacji wody, której lepkość nie jest taka duża, jak np. miodu) możemy sobie pozwolić na całkowite pominięcie algorytmu dyfuzji. Opis algorytmu dyfuzji wraz z implementacją zawarty jest w pracy J. Stama [18].

2.3.3 Ekstrapolacja

Wewnątrz płynu założenie o jednorodności pola prędkości jest dobre. Problem pojawia się jednak na powierzchni płynu, gdzie założenie to nie jest już poprawne. Granica pomiędzy wodą a powietrzem wyróżnia się dużą różnicą prędkości. W powietrzu prędkość może wynosić zero, a w związku z tym, ponieważ wykonujemy adwekcję wstecz, nigdy nie

przemieścimy się i nie trafi do komórki z powietrzem. W ten sposób natrafimy na barierę, która sprawi, że woda nie będzie w stanie się przemieścić (patrz eksperyment 3.5.2).

Sposobem na rozwiązanie tego problemu jest wykonywanie ekstrapolacji pola prędkości w powietrzu. Polega to na tym, że tuż przed algorytmem adwekcji sztucznie ustawiamy prędkość w powietrzu na wartość znajdująca się w najbliższej sąsiadujących komórkach wody. Więcej informacji na temat ekstrapolacji można znaleźć w notatkach [4].

2.4 Algorytm projekcji

Algorytm projekcji łączy ze sobą część równania ruchu, warunek nieściśliwości i warunki brzegowe:

$$\vec{u}_{n+1} = \vec{u}_n - \frac{1}{\rho} \nabla p \Delta t \quad (2.14)$$

$$\nabla \cdot \vec{u}_{n+1} = 0 \quad (2.15)$$

$$\vec{u}_{n+1} \cdot \hat{n} = 0 \quad (2.16)$$

Pierwszą rzeczą, jaką należy zrobić, jest rozpisanie zdyskretyzowanych równań na ciśnienie. Zapisujemy więc kolejne składowe wektora prędkości w równaniu (2.14) i dyskretyzujemy wyraz ∇p . Warto wrócić do Rysunku 2.1 który pojawił się wcześniej w celu zobrazowania dyskretyzacji.

$$u_{i+1/2,j,k}^{n+1} = u_{i+1/2,j,k} - \Delta t \frac{1}{\rho} \frac{p_{i+1,j,k} - p_{i,j,k}}{\Delta x} \quad (2.17)$$

$$v_{i,j+1/2,k}^{n+1} = v_{i,j+1/2,k} - \Delta t \frac{1}{\rho} \frac{p_{i,j+1,k} - p_{i,j,k}}{\Delta x} \quad (2.18)$$

$$w_{i,j,k+1/2}^{n+1} = w_{i,j,k+1/2} - \Delta t \frac{1}{\rho} \frac{p_{i,j,k+1} - p_{i,j,k}}{\Delta x} \quad (2.19)$$

Tu pojawia się właśnie powód, dla którego prędkość trzymana jest na ścianach sześciianów zamiast w ich centrach. Gdybyśmy trzymali prędkości w centrach, gradient $\nabla \rho$ wymagałby od nas mniej dokładnej dyskretyzacji.

Następnie zdyskretyzujemy warunek nieściśliwości:

$$\nabla \cdot \vec{u}_{n+1} = 0 \quad (2.20)$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \quad (2.21)$$

Spoglądając znowu na MAC grid widzimy, że wynikiem dyskretyzacji jest:

$$\frac{u_{i+1/2,j,k}^{n+1} - u_{i-1/2,j,k}^{n+1}}{\Delta x} + \frac{w_{i,j+1/2,k}^{n+1} - v_{i,j-1/2,k}^{n+1}}{\Delta x} + \frac{w_{i,j,k+1/2}^{n+1} - w_{i,j,k-1/2}^{n+1}}{\Delta x} = 0 \quad (2.22)$$

Łącząc równania (2.17) i (2.22) otrzymujemy po przekształceniach:

$$\begin{aligned} & 6p_{i,j,k} - p_{i+1,j,k} - p_{i,j+1,k} - p_{i,j,k+1} - p_{i-1,j,k} - p_{i,j-1,k} - p_{i,j,k-1} = \\ & = \frac{\rho \Delta x}{\Delta t} (u_{i-1/2,j,k} + v_{i,j-1/2,k} + w_{i,j,k-1/2} - u_{i+1/2,j,k} - v_{i,j+1/2,k} - w_{i,j,k+1/2}) \end{aligned} \quad (2.23)$$

Taką postać równanie to przybiera, gdy komórka MAC grida sąsiaduje z komórkami wody. Jeśli sąsiaduje ona z komórką powietrza, to odpowiedni wyraz ciśnienia p wynosi zero, tak jak wspomniane było w rozdziale 1.4. Jeśli natomiast rozważana komórka sąsiaduje z komórką ściany, to wzór (2.17) w połączeniu z (2.16) przyjmuje postać:

$$0 = u_{i+1/2,j,k} - \Delta t \frac{1}{\rho} \frac{p_{i+1,j,k} - p_{i,j,k}}{\Delta x} \quad (2.24)$$

$$p_{i+1,j,k} = p_{i,j,k} + \frac{\rho \Delta x}{\Delta t} u_{i+1/2,j,k} \quad (2.25)$$

Wstawiając do wzoru (2.23) i przekształcając otrzymujemy:

$$\begin{aligned} 5p_{i,j,k} - p_{i,j+1,k} - p_{i,j,k+1} - p_{i-1,j,k} - p_{i,j-1,k} - p_{i,j,k-1} = \\ = \frac{\rho \Delta x}{\Delta t} (u_{i-1/2,j,k} + v_{i,j-1/2,k} + w_{i,j,k-1/2} - 0 - v_{i,j+1/2,k} - w_{i,j,k+1/2}) \end{aligned} \quad (2.26)$$

Jak widać, gdy mamy do czynienia z warunkiem brzegowym ściany, to wartość przy $p_{i,j,k}$ maleje, usuwana jest z równania wartość ciśnienia p odpowiadająca ścianie i prędkość przy ścianie ustawiana jest na zero. Gdy mamy do czynienia z warunkiem brzegowym powietrza, to usuwana jest z równania wartość ciśnienia p odpowiadająca powietrzu.

2.4.1 Zebranie w postać macierzową

Równań postaci takiej jak (2.23) otrzymamy tyle, ile jest komórek wody w naszej siatce MAC grid. Teraz należy taki układ równań rozwiązać. Najlepszym sposobem jest przedstawienie tego układu równań w postaci równania macierzowego:

$$Ap = d \quad (2.27)$$

Symbol p to wektor ciśnień $p_{i,j,k}, p_{i+1,j,k}, \dots$, który jest przez nas szukany. Macierz A zawiera na przekątnej wartości będące liczbą komórek sąsiadujących (niebędących ścianami), a w pozostałych elementach każdego wiersza do sześciu wartości -1 . Wartości

te odpowiadają współczynnikom zmiennych z lewej strony równania (2.23). Symbol d to wektor dywergencji, czyli prawa strona równania (2.23).

Aby rozwiązać taki układ równań nie możemy niestety posłużyć się **eliminacją Gausa**, gdyż ma ona zbyt dużą złożoność obliczeniową. Na szczęście macierz A jest macierzą symetryczną dodatnio określona, dzięki czemu możemy posłużyć się metodami aproksymacyjnymi, takimi jak: **metoda Jacobiego**, **metoda Gaussa-Seidla** [8] lub **metoda gradientu sprzężonego (conjugate-gradient)** [13], [4].

2.4.2 Nazwa „projekcja”

Ciekawostką dotyczącą algorytmu projekcji jest fakt, iż działa on jak operacja projekcji w algebrze liniowej. Operacja projekcji wykonana ponownie na wyniku projekcji daje ten sam wynik, tj. projekcja to taka funkcja P , że $P(P(x)) = P(x)$ [11]. I co ciekawe, algorytm projekcji działa dokładnie tak samo. To jest właśnie powodem, dla którego algorytmy symulacji są wykonywane szeregowo zamiast równolegle.

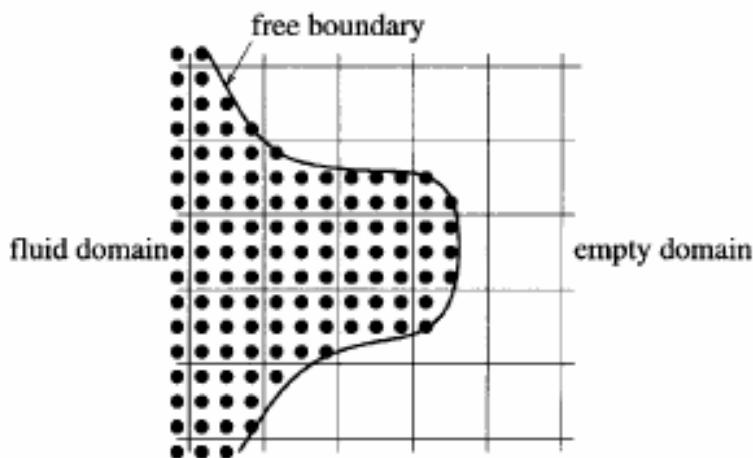
2.5 Reprezentacja płynu

Na koniec, gdy znane są już nam wszystkie algorytmy, oraz sposób reprezentacji danych w siatce MAC grid, pozostaje jeszcze tylko jedna rzecz. W jaki sposób reprezentować płyn? W jaki sposób odróżnić gdzie w danym momencie symulacji znajduje się woda, a gdzie powietrze? Odpowiedzi na te pytania są przynajmniej trzy.

Pierwsza to **pole gęstości**, wykorzystane np. w pracy [18]. Metoda ta polega, na zapisywaniu w centrach komórek MAC grida wartość gęstości. Podlega ona adwekcji podczas symulacji, podobnie jak prędkość. Gdy wartość gęstości jest większa od pewnej wartości progowej w danej komórce, to jest tam woda, jeśli jest mniejsza, to jest tam powietrze.

Druga metoda, obecnie najpopularniejsza, to tzw. **level sets**, opisane dokładnie w pracach [16] [17] [4]. Metoda ta jest bardzo podobna do pola gęstości, z tym, że zamiast zapisywania ile płynu znajduje się w danej komórce, zapisywana jest najbliższa odległość danej komórki do powierzchni płynu. Dla komórek powietrza wartość ta jest dodatnia, a dla komórek wody wartość ta jest ujemna. Dokonując adwekcji takiego pola (zwane z angielskiego **signed distance**), możemy nie tylko stwierdzić, czy w danej komórce znajduje się woda czy powietrze, ale także jak bardzo oddalona jest ona od powierzchni, by w ten sposób uzyskiwać lepszy efekt wizualny symulacji.

Trzecia metoda polega na zastosowaniu tzw. **markerów częstek**. Ta metoda została wykorzystana w niniejszej pracy. Łączy ona ze sobą podejście Lagrange'a i Eulera. Polega ona na tym, iż w komórkach, w których występuje woda, tworzy się tzw. markery (częstki wody), które następnie podlegają adwekcji, tak jak prędkość czy gęstość. Dzięki nim można jednoznacznie stwierdzić, gdzie znajduje się woda, a gdzie jej nie ma. W przeciwieństwie do metod wcześniejszych, markery podlegają adwekcji do przodu, a nie wstecz, i nie cierpią na problemy związane z dyssypacją. W zamian za to, wymagają wykonywania dodatkowych obliczeń i dają nieco gorsze efekty wizualne. Opisane są także w książce [12].



Rysunek 2.4: Markery częstek [12]

Wprawdzie pojedynczy marker nie cierpi bezpośrednio na dyssypację, ale w praktyce

jednak markery jako grupa niekoniecznie zachowują objętość (patrz Rozdział 3.4). Może się zdarzyć, że początkowo wstawimy na każdą komórkę 9 markerów tam, gdzie jest woda, ale po pewnym czasie symulacji w jednej komórce markerów może znaleźć się ich więcej, a w innej mniej. W ten sposób objętość płynu nie zostanie zachowana.

2.6 Schemat symulacji

W pracy tej porzucony został algorytm dyfuzji, dodane zostały za to algorytm ekstrapolacji i mechanizm markerów, w związku z tym ostatecznie użyty w pracy wariant schematu symulacji wygląda następująco:

1. Ustaw warunki początkowe: pole \vec{u}_0
2. Wybierz odpowiedni krok Δt
3. Wykonaj algorytm ekstrapolacji prędkości
4. Wykonaj algorytm adwekcji do przodu markerów
5. Wykonaj algorytm adwekcji wstecz prędkości: $\vec{u}_{n+1} = A(u_n, \Delta t)$
6. Wykonaj algorytm sił zewnętrznych: $\vec{u}_{n+3} = G(u_{n+2}, \Delta t)$
7. Algorytm projekcji: $\vec{u}_{n+4} = P(u_{n+3}, \Delta t)$

Powtarzaj kroki 2 – 7.

Rozdział 3

Eksperymenty

W celu przetestowania skuteczności wcześniej opisanych metod numerycznych symulacji mechaniki płynów, wykonano ich implementację oraz przeprowadzono eksperymenty. Każdy z eksperymentów został przeprowadzony dla różnych parametrów. Wszystkie były przeprowadzone na siatce MAC grid wielkości $40 \times 40 \times 40$. W dalszej części tego rozdziału eksperymenty te zostaną dokładnie opisane.

3.1 Woda stojąca

Pierwszym eksperymentem jest symulacja stojącej wody. Stworzone zostało naczynie w kształcie sześciangu, całkowicie wypełnione wodą. Woda w żaden sposób się nie porusza i całość stoi nieruchomo. Na pierwszy rzut oka nie widać w tym niczego interesującego, niemniej jednak jest to bardzo ważny eksperiment, który pozwala na uzyskanie wielu ciekawych informacji.

3.1.1 Wydajność

Istotnym spostrzeżeniem jest, że wbrew pozorom jest to jeden z najbardziej kosztownych obliczeniowo przypadków do symulacji. Jest to spowodowane tym, iż prawie wszystkie komórki siatki MAC grid są wypełnione wodą, a więc algorytm projekcji, który jest najbardziej wymagający obliczeniowo, musi wykonać niemal maksymalnie dużo obliczeń. Można zaobserwować więc, biorąc pod uwagę liczbę klatek na sekundę, że jest to przypadek bardziej wymagający niż kolejne, opisane w dalszej części rozdziału. Dlatego też jest

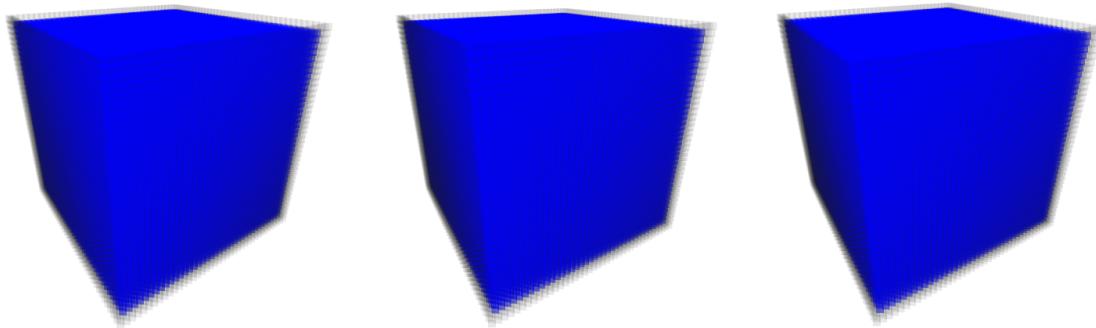
to dobry eksperyment, pozwalający na badanie wydajności naszej implementacji oraz pozwalający nam na jej poprawianie.

Porównując ze sobą kolejne zaimplementowane algorytmy rozwiązywania równań, uzyskano następujące wyniki:

FPS: 6.224693
Czas: 0.490000 s
Max P: 3720.976562 hPa

FPS: 10.045910
Czas: 0.640000 s
Max P: 3724.107910 hPa

FPS: 56.351150
Czas: 5.990042 s
Max P: 3724.035889 hPa



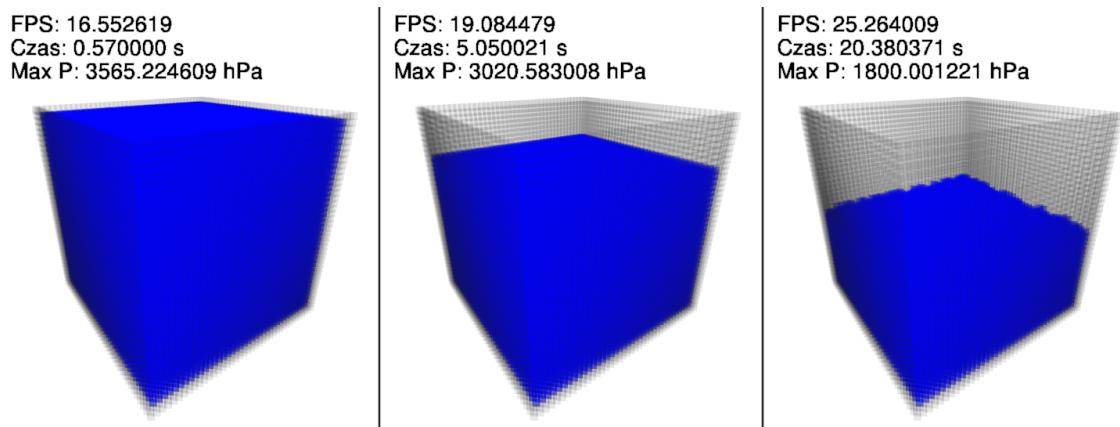
Rysunek 3.1: Wydajność algorytmów. Od lewej: Gaussa-Seidla, Conjugate-Gradient, Conjugate-Gradient prekondycjonowany wynikami z poprzedniego kroku

Jak można więc zauważyć, najwydajniejszy jest algorytm gradientu sprzężonego (ang. *preconditioned conjugate-gradient*).

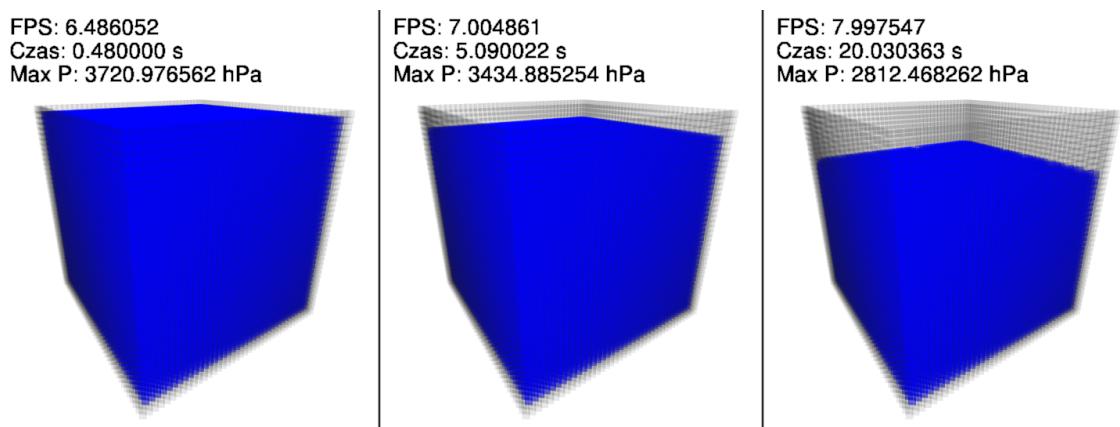
3.1.2 Zbieżność metod rozwiązywania układów równań

Kolejną rzeczą, która jest poddawana próbie w tym eksperymencie, jest poprawność i szybkość zbiegania metod aproksymacyjnych rozwiązywania układów równań. Metody aproksymacyjne, takie jak metoda Gaussa-Seidla lub metoda gradientu sprzężonego (z ograniczoną liczbą iteracji), dają nam jedynie przybliżony wynik poprawnego rozwiązania. Dlatego też oczywistym jest, że dla zbyt małej liczby iteracji otrzymane przybliżenie nie będzie wystarczająco dokładne, a przybliżone wyniki sprawią, że nasza symulacja nie będzie się zachowywać tak jakbyśmy tego oczekiwali.

Badając więc odpowiednio algorytmy, otrzymano takie oto wyniki:



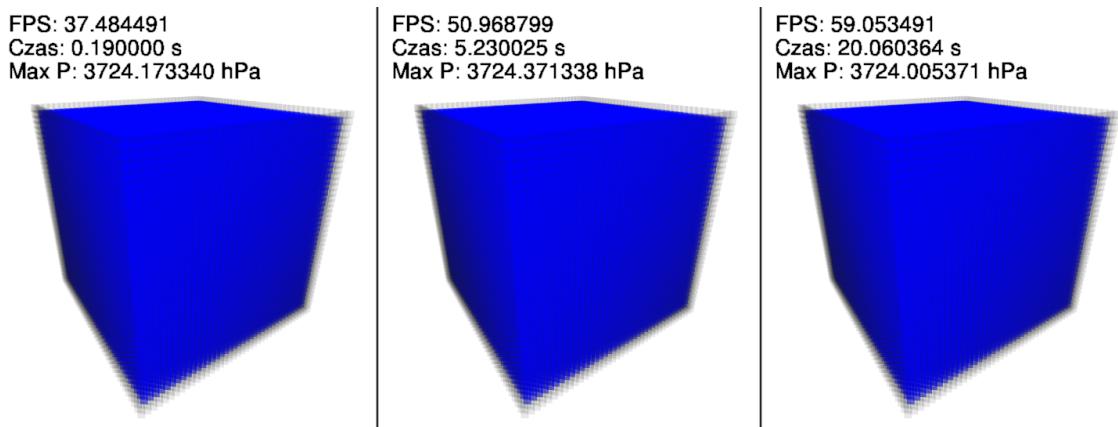
Rysunek 3.2: Algorytm Gaussa-Seidla, liczba iteracji: 100



Rysunek 3.3: Algorytm Gaussa-Seidla, liczba iteracji: 300



Rysunek 3.4: Algorytm Conjugate-Gradient, liczba iteracji: 20



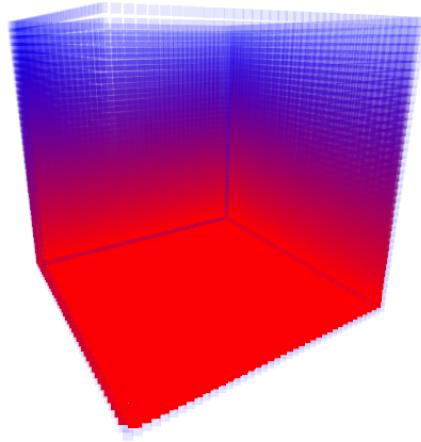
Rysunek 3.5: Algorytm Conjugate-Gradient, liczba iteracji: 80

Jak można zauważyć, gdy liczba iteracji jest zbyt mała, to ciśnienie jest źle obliczane i woda się „ścisza”, przez co jej ubywa. Jedynie metoda gradientu sprzężonego z liczbą iteracji równą co najmniej 80 wystarczyła by wynik był wystarczająco dokładny i by nie doszło do niepożądanego ubywania wody. Dodatkowo można zaobserwować, że metoda gradientu sprzężonego zbiega zdecydowanie szybciej (wymaga mniejszej liczby iteracji) od metody Gaussa-Seidla.

3.1.3 Poprawność obliczeń

Ostatnią ciekawą rzeczą, którą można zbadać przy użyciu takiego eksperymentu, jest poprawność obliczania ciśnienia hydrostatycznego. Ciśnienie hydrostatyczne wody stojącej można łatwo obliczyć ze wzoru $P = \rho gh$. Gęstość ρ wynosi $1000 \frac{kg}{m^3}$, przyspieszenie ziemskie $g = 9.8 \frac{m}{s^2}$, a wysokość słupa wody $h = 38m$. Ciśnienie na dnie powinno więc wynosić $P = 3724hPa$. Można więc zauważyć, że nasz program faktycznie taki wynik podaje, co jest bardzo dobrą przesłanką na to, że algorytm projekcji działa poprawnie (gdyby takiego wyniku nie podawał z pewnością działałby niepoprawnie). Dodatkowo można zauważyć na Rysunku 3.6 poniżej, że ciśnienie jest większe w dolnych częściach naczynia i mniejsze w górnych (kolor czerwony - duże ciśnienie, kolor niebieski - niskie ciśnienie).

FPS: 59.661837
Czas: 9.720127 s
Max P: 3724.012451 hPa



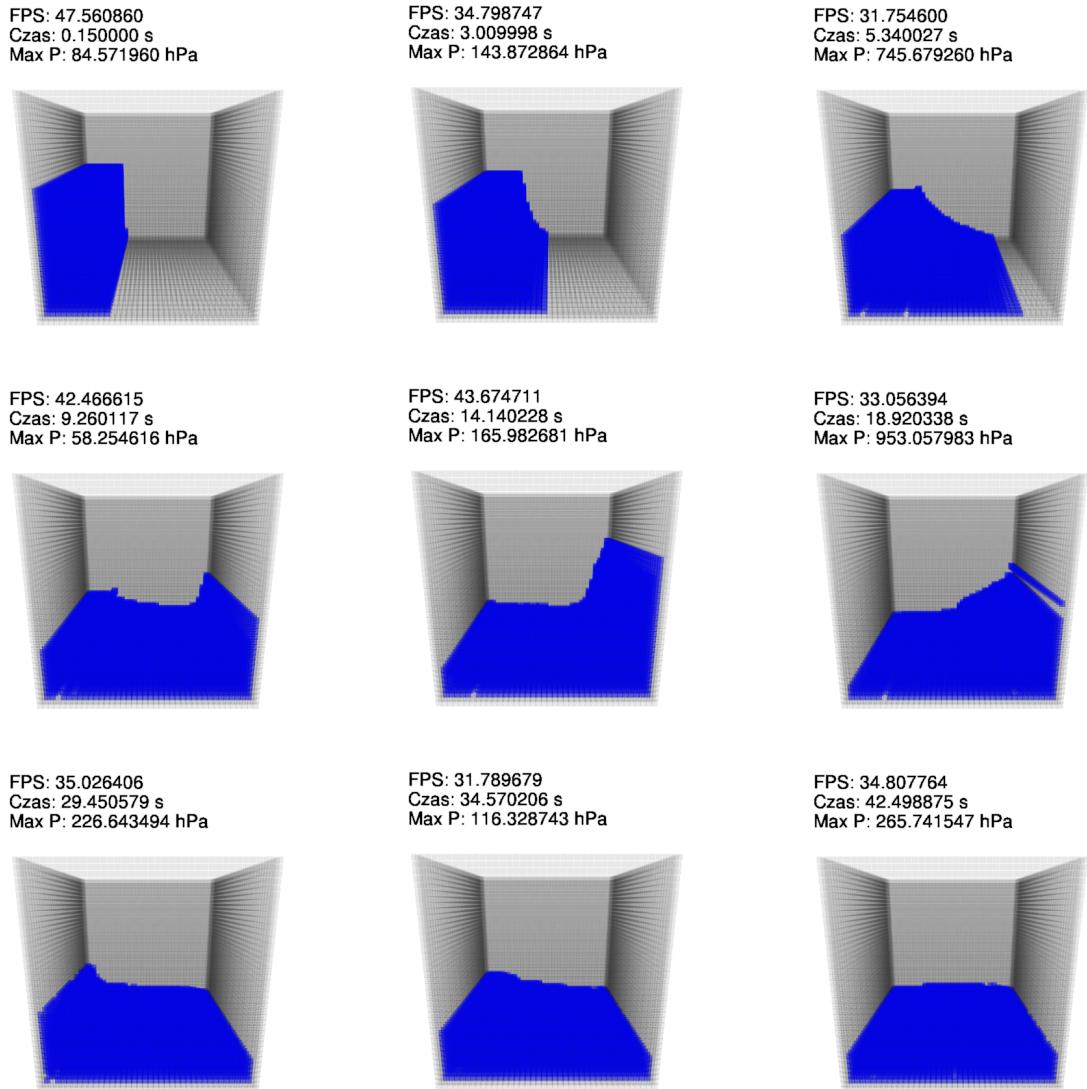
Rysunek 3.6: Ciśnienie wywierane na ściany naczynia

3.2 Falowanie wody

Drugim przeprowadzonym eksperymentem jest falowanie wody. Polegał on na wypełnieniu sześciennego naczynia w jednej części wodą, a w drugiej części pozostawieniu go pustego. Powinniśmy się spodziewać, że pod wpływem siły grawitacji woda przeleje się na drugą stronę, następnie odbije się od ściany i wróci z powrotem. Woda będzie się tak odbijać aż ostatecznie się ustabilizuje i uspokoi na dnie naczynia.

3.2.1 Wizualizacja

Skoro wiemy czego powinniśmy się spodziewać po danym eksperymencie warto sprawdzić czy faktycznie otrzymamy wynik którego oczekiwaliśmy. Wykonany został więc eksperyment z parametrem $\Delta t = 0.01s$ i otrzymano taki efekt wizualny, jak na Rysunku 3.7:



Rysunek 3.7: Wizualizacja eksperymentu falującej wody

Efekt wizualny okazał się więc być taki jak przewidywaliśmy.

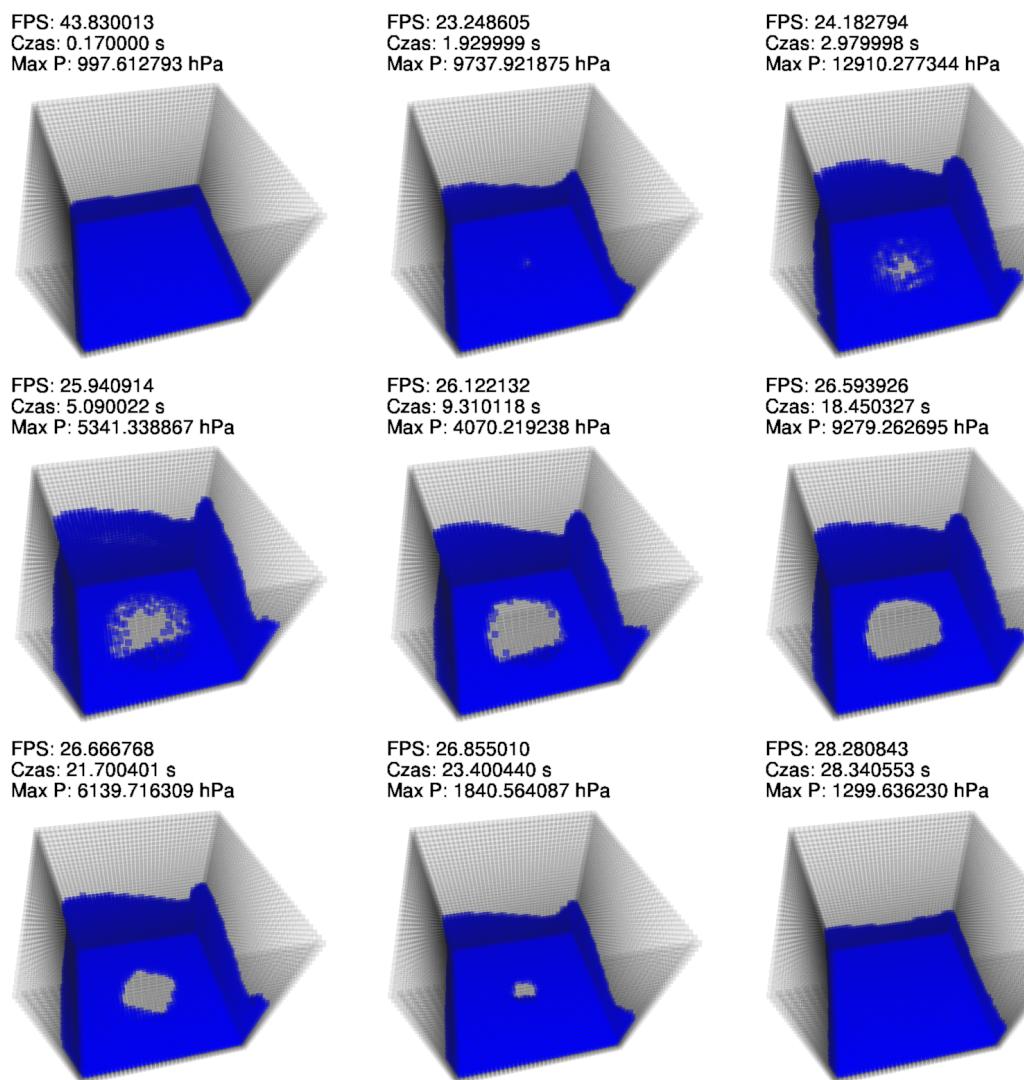
3.3 Ruch wirowy

Trzecim eksperymentem jest ruch wody po okręgu. Eksperyment ten polega na wypełnieniu dna sześciennego naczynia wodą, a następnie na wprawieniu wody w ruch wirowy.

Spodziewalibyśmy się więc ujrzeć nie tylko ruch wirowy wody, ale także wpływ siły odśrodkowej, a co za tym idzie, na brzegach naczynia powinno znaleźć się więcej wody niż w środku. Na koniec siła wprawiająca wodę w ruch jest wyłączana i można ujrzeć jak woda z powrotem powraca do pierwotnego stanu.

3.3.1 Wizualizacja

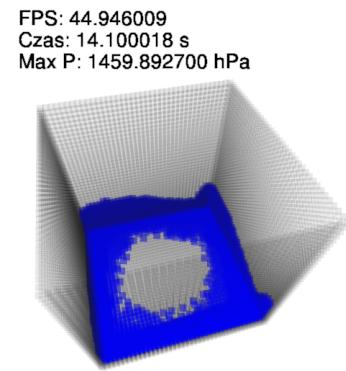
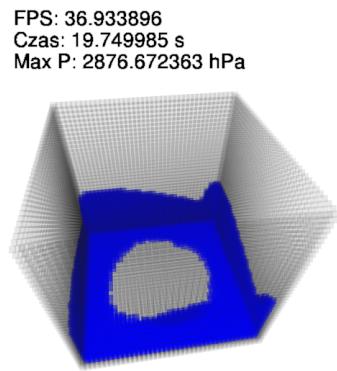
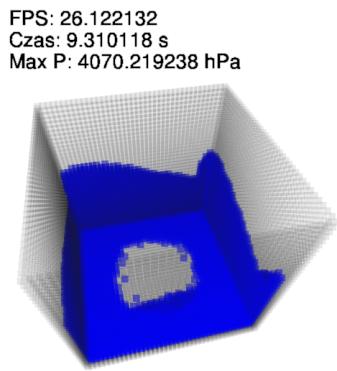
Przykład symulacji ruchu wirowego został zilustrowany na Rysunku 3.8.



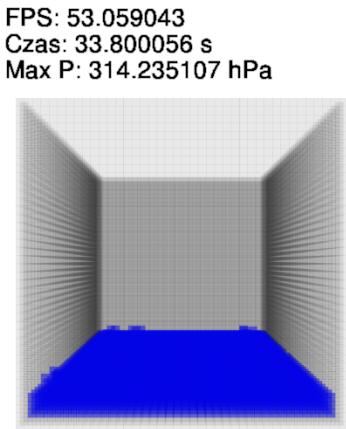
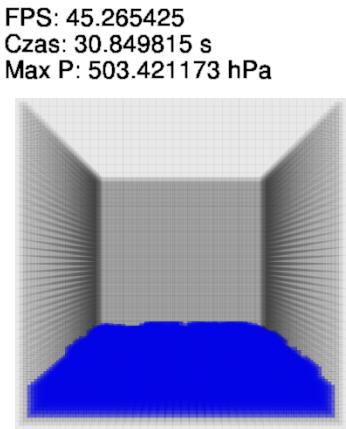
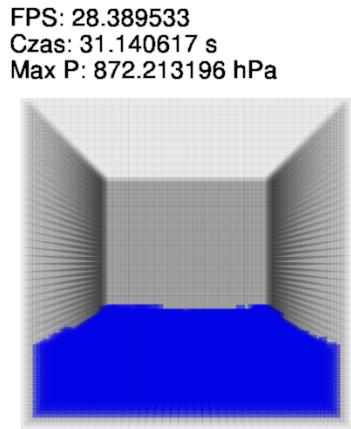
Rysunek 3.8: Wizualizacja eksperymentu wirującej wody, $\Delta t = 0.01\text{s}$

3.3.2 Dyssypacja

Ruch wirowy wody jest doskonałym sposobem na sprawdzenie praktycznego wpływu dyssypacji na symulację. Wynika to z faktu, iż ruch po okręgu jest najbardziej „podatny” na wszelkiego rodzaju dyfuzję. Dodatkowo też wiemy, że dyssypacja jest tym większa, im większy jest krok czasowy. Dlatego też przeprowadzony został ten eksperyment dla różnych wartości Δt . Otrzymane wyniki znajdują się na Rysunkach 3.9 i 3.10.



Rysunek 3.9: Wizualizacja wiru. Od lewej: $\Delta t = 0.01s$, $\Delta t = 0.05s$, $\Delta t = 0.1s$



Rysunek 3.10: Woda po powrocie do stanu pierwotnego. Od lewej: $\Delta t = 0.01s$, $\Delta t = 0.05s$, $\Delta t = 0.1s$

Z rysunków można zauważyć, że im większy krok czasowy (im większa dyssypacja) tym woda bardziej się „przyciska” do ścian (dziura na środku jest większa), a na końcu

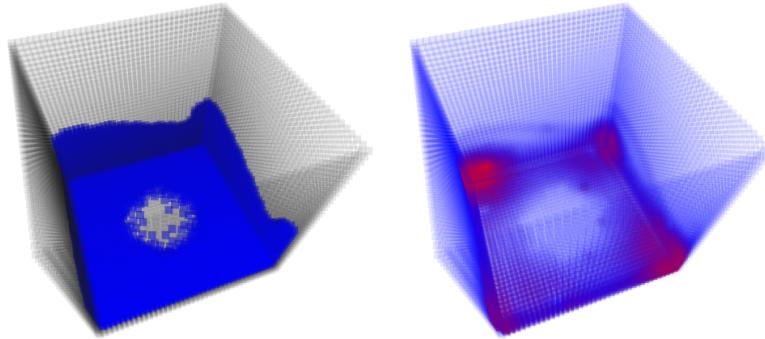
symulacji, po powrocie do stanu pierwotnego, wody ubywa tym więcej im większa była dyssypacja. Ważne jest więc, by krok czasowy był możliwie jak najmniejszy, aby objętość wody była zachowana.

3.3.3 Ciśnienie

Ruchowi po okręgu towarzyszy siła odśrodkowa, a wraz z rosnącą siłą rośnie także ciśnienie. Eksperyment pozwala na sprawdzenie jak duże ciśnienie wystąpi w takim przypadku w różnych obszarach symulowanego płynu.

FPS: 27.792887
Czas: 8.930109 s
Max P: 3138.081055 hPa

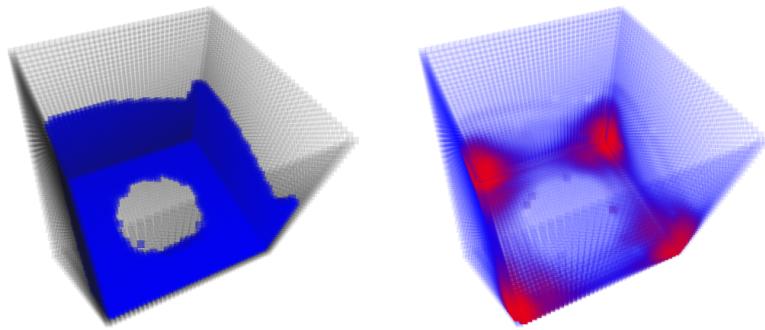
FPS: 27.792887
Czas: 8.930109 s
Max P: 3138.081055 hPa



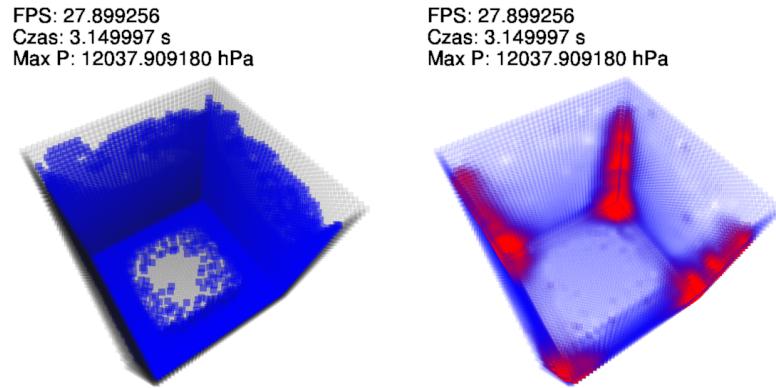
Rysunek 3.11: Z lewej woda, z prawej ciśnienie dla prędkości stycznej równej $5 \frac{m}{s}$

FPS: 26.055625
Czas: 11.500168 s
Max P: 4490.855469 hPa

FPS: 26.055625
Czas: 11.500168 s
Max P: 4490.855469 hPa



Rysunek 3.12: Z lewej woda, z prawej ciśnienie dla prędkości stycznej równej $10 \frac{m}{s}$



Rysunek 3.13: Z lewej woda, z prawej ciśnienie dla prędkości stycznej równej $20\frac{m}{s}$

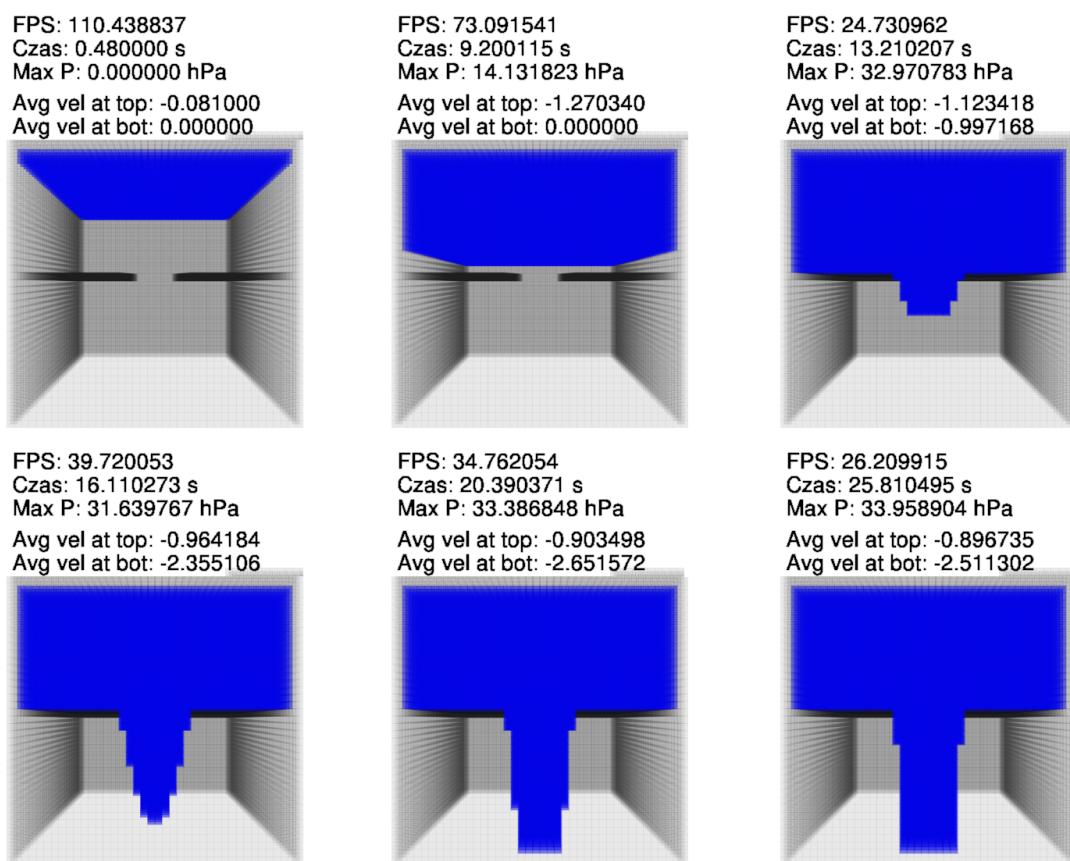
Jak widać, rzeczywiście im większa jest prędkość ruchu wody, tym większa jest siła odśrodkowa i tym większe jest ciśnienie działające na ściany naczynia. Dodatkowo można zauważyc, że największe ciśnienie występuje w rogach naczynia, co może być bardzo interesujące, szczególnie z punktu widzenia inżyniera, który chciałby zaprojektować naczynie (wybrać odpowiedni materiał), tak aby wytrzymało ono nacisk wody.

3.4 Zwężeńie

Czwartym eksperymentem jest zwężeńie. Polega on na sprawdzeniu nieściśliwości wody, podczas zmniejszania pola przekroju, przez które woda przepływa. Stworzone zostało naczynie, przez które przepływa woda. W połowie drogi napotyka ona przeszkodę w postaci zwężenia. Obserwujemy jak woda w tym zwężeniu przyspiesza.

3.4.1 Wizualizacja

Przykład symulacji wody przepływającej przez zwężeńie został przedstawiony na Rysunku 3.14.



Rysunek 3.14: Wizualizacja ruchu wodą przez zwężeńie

3.4.2 Nieściśliwość wody

Jak można zauważyc podczas wizualizacji, woda w dolnej części naczynia faktycznie porusza się szybciej niż w górnej, co jest zrozumiałe, gdyż aby woda nie zmieniła swojej objętości musi przyspieszyć – by ilość wody, która wpływa była równa ilości wody, która wypływa z naczynia.

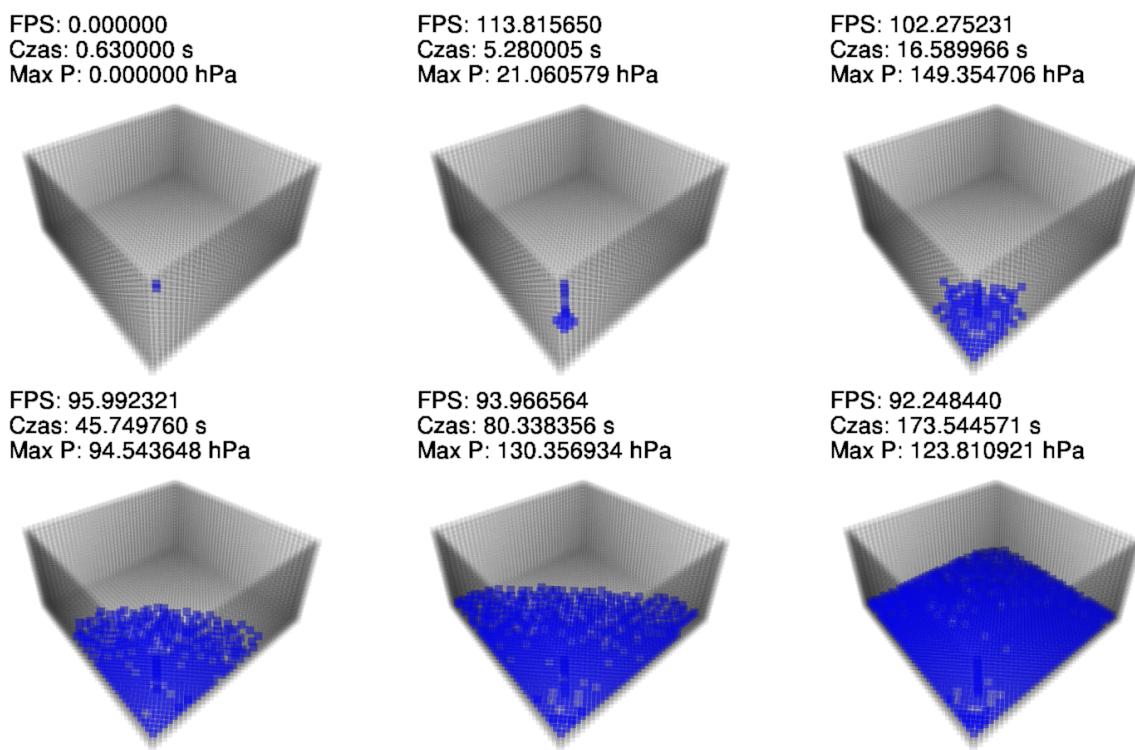
Niemniej jednak, pole powierzchni zwężenia jest cztery razy mniejsze od pola powierzchni wlotowej wody. W związku z tym, powinniśmy uzyskać około czterokrotne przyspieszenie, a uzyskano jedynie około 2,8 krotne. Zauważyc możemy w tym momencie niedoskonałość naszego algorytmu projekcji. Jest to spowodowane faktem, iż zaimplementowany algorytm projekcji nie jest uzależniony od gęstości i przy bardzo dużych różnicach gęstości nie zachowuje on całkowicie objętości płynu. Sposób na poprawienie tego problemu został opisany w ostatnim rozdziale 4.

3.5 Woda lejąca się z kranu

Ostatnim eksperymentem jest rozlewanie się wody ze źródła, które mogłoby być kranem umieszczonym nad naszym naczyniem. Eksperyment ten wizualnie powinien zaprezentować w jaki sposób woda równomiernie rozprowadza się po całym naczyniu. Co ciekawe z eksperymentem tym można było natrafić na różne problemy, które zostaną w dalszej części opisane.

3.5.1 Wizualizacja

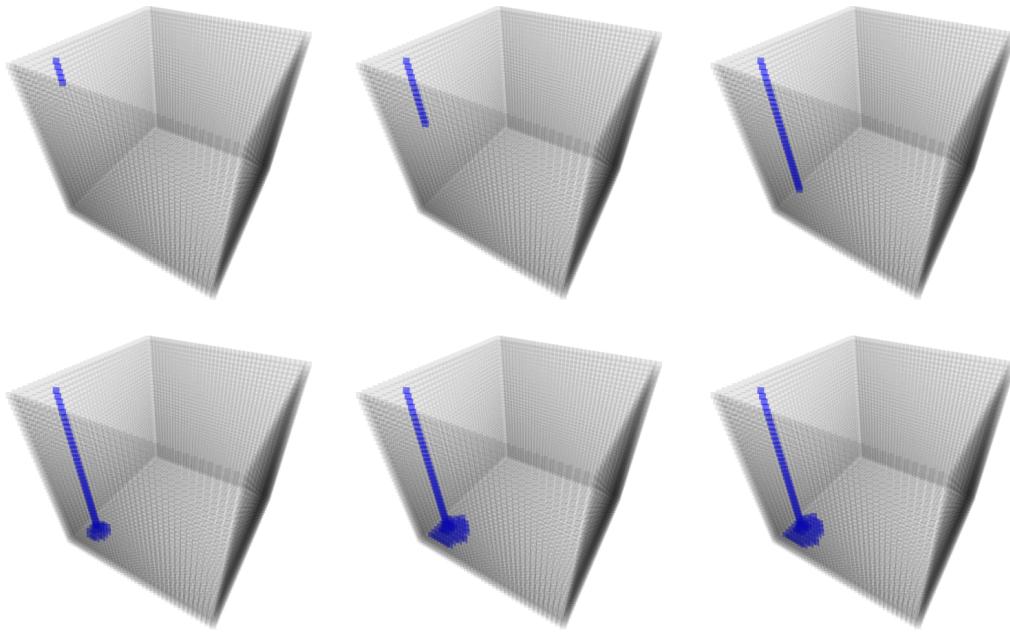
Przykład symulacji rozlewającej się wody został zilustrowany na Rysunku 3.15.



Rysunek 3.15: Wizualizacja rozlewającej się wody

3.5.2 Problem zatrzymania wody

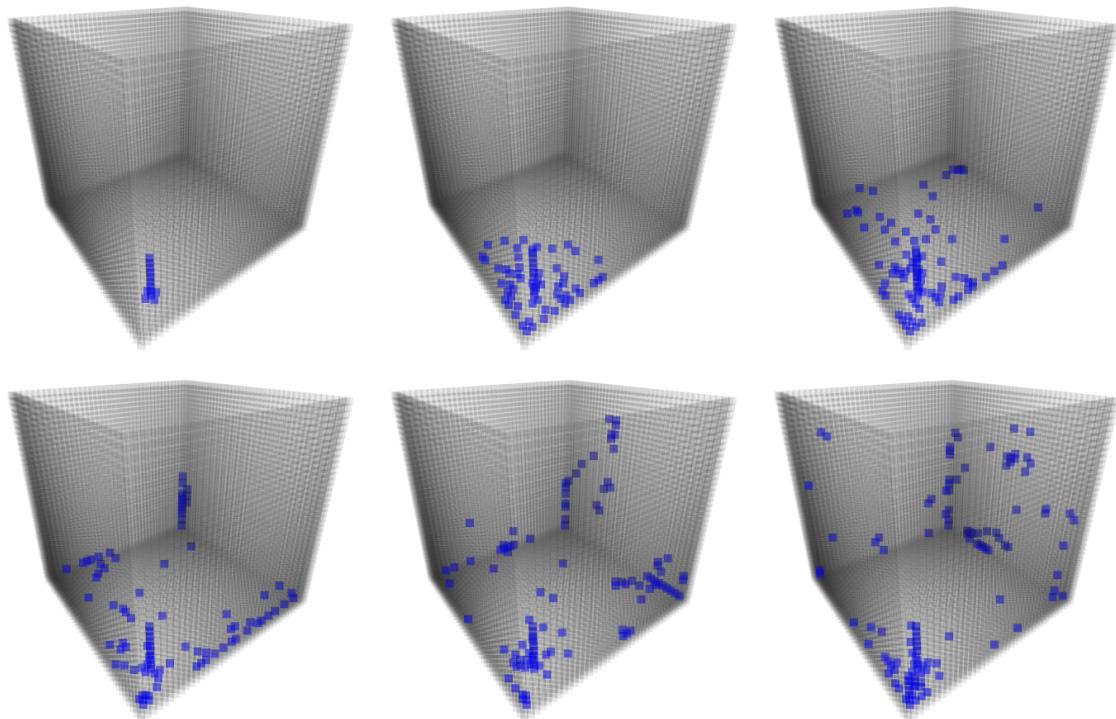
Ciekawym problemem, na który można było się natknąć podczas przeprowadzania tego eksperymentu, był fakt, iż woda w dziwny sposób zastygała w miejscu. Dochodziła ona do pewnej granicy, której nie mogła już przekroczyć. Jak się okazało, było to spowodowane brakiem algorytmu ekstrapolacji i dyssypacja prędkości, co sprawiało, że prędkość w granicy wynosiła zero. Dowodzi to faktu, iż algorytm ekstrapolacji jest praktycznie niezbędny przy symulacji mechaniki płynów.



Rysunek 3.16: Wizualizacja rozlewającej się wody napotykającej na granicę

3.5.3 Problem braku oporu

Drugi problem, jaki mógł wystąpić, jest w pewnym sensie przeciwnieństwem poprzedniego. W tym przypadku woda nie tylko nie wpada na granicę, której nie może przekroczyć, ale wręcz płynie w nieskończoność, nawet przeciwnie do siły grawitacji, po czym wydostaje się poza symulowany obszar. Jest to spowodowane faktem, iż na wodę nie działa żadna siła oporu. W związku z tym symulowana woda rozpędza się i po pewnym czasie opuszcza nasze naczynie. Należy zauważyć jednak, że w rzeczywistej próżni takie zjawisko również miałoby miejsce. Sposobem na rozwiązanie tego problemu jest dodanie siły oporu powietrza, która będzie trzymać prędkość wody w ryzach. W praktyce można jednak problem ten rozwiązać w inny sposób. Można wykorzystać tym celu dyssypację. Ponieważ dyssypacja prędkości sama sprawia, że prędkość będzie maleć, przy odpowiednich parametrach można więc uzyskać efekt taki, jak gdyby istniał opór powietrza – będzie on „symulowany” przez dyssypację. Efekt ten został zilustrowany na Rysunku 3.17



Rysunek 3.17: Wizualizacja rozlewającej się wody napotykającej na granicę

3.6 Wnioski

Podsumowując, z przeprowadzonych eksperymentów możemy wywnioskować następujące stwierdzenia:

- Metoda gradientu sprzężonego, zastosowana w algorytmie projekcji, daje najlepsze wyniki oraz jest najwydajniejsza obliczeniowo.
- Dyssypacja faktycznie jest tym większa, im większy jest krok czasowy Δt .
- Szczególnie podatny na dyssypację jest ruch po okręgu.
- Algorytm ekstrapolacji jest niezbędny do poprawnego symulowania mechaniki pływów.

- Dyssypacja jest na ogół niepożądana, ale jednak czasami możemy wykorzystać ją na własną korzyść.

Rozdział 4

Podsumowanie

Podsumowując przeprowadzone na potrzeby pracy badania, można zauważyc, że symulacje komputerowe mechaniki płynów są tematem bardzo obszernym. Do symulacji takiej wykorzystuje się wiele algorytmów, które złożone w całość potrafią stworzyć bardzo dobrą wizualizację oraz dostarczyć wielu cennych informacji. Dodatkowo, w dalszym ciągu prowadzone są prace nad udoskonalaniem tychże algorytmów oraz opracowywane są nowe metody, realizujące wymagane elementy symulacji coraz dokładniej i wydajniej. Praca ta więc w żaden sposób nie wyczerpuje tematu i można obrać wiele dalszych kierunków pracy.

4.1 Reprezentacja płynu

Pierwszym z miejsc, w których można by było poczynić kolejne kroki, jest przetestowanie różnych reprezentacji płynu. W pracy tej wykorzystane w tym celu zostały markery, więc dalszym kierunkiem pracy mogłoby być wypróbowanie pola gęstości oraz metody *level set*, które są szeroko opisane w literaturze, m.in. w pracach [16], [17], [4]. Być może dałyby one lepszy efekt wizualny oraz dokładniejsze wyniki dla wybranych eksperymentów.

4.2 Algorytm adwekcji

Kolejnym miejscem, w którym można prowadzić dalsze badania jest udoskonalanie algorytmu adwekcji. Istnieją metody, które pozwalają na zwalczanie dyssypacji, takie jak „*vorticity confinement*”, opisane m.in. w [4] albo [5].

4.3 Algorytm projekcji

Elementem, który zdecydowanie można udoskonalić, jest algorytm projekcji. Jak już wcześniej było to pokazane na przykładzie eksperymentu zwężenia 3.4, algorytm ten zaimportowany na potrzeby tej pracy nie jest doskonały. Pierwszym krokiem ku udoskonaleniu tego algorytmu byłoby uzależnienie go od gęstości, co jest opisane w pracach [3] [7]. Dodatkowo, można w tym miejscu zastosować lepszy algorytm rozwiązywania układów równań liniowych, na przykład wykorzystując **Incomplete Cholesky Decomposition** [4] oraz zrównoleglając obliczenia.

Warto także w tym miejscu zaznaczyć, że implementacja zastosowana w tej pracy wykonuje wszystkie obliczenia numeryczne na procesorze głównym (CPU). Jednym z najbardziej obiecujących sposobów poprawy wydajności byłoby przepisanie implementacji w taki sposób, by obliczenia były wykonywane również przez procesor graficzny (GPU).

4.4 Mieszanie płynów

Jeszcze jednym z dalszych kierunków prac mogłoby być stworzenie symulacji, w której symulowany jest więcej niż jeden rodzaj płynu. Przykładowo, można by w ten sposób symulować równocześnie ruchy powietrza oraz wody i stworzyć symulację ruchu fal morskich. Wymagałoby to stworzenia odpowiedniego algorytmu projekcji oraz odpowiedniego śledzenia powierzchni różnych płynów [7].

Bibliografia

- [1] *open-gl tutorial.* <http://www.opengl-tutorial.org>.
- [2] *OpenGL Tutorial.* <https://open.gl>.
- [3] John B. Bell and Daniel L. Marcus. *A Second-Order Projection Method for Variable-Density Flows.* Journal of Computational Physics, Livermore, California, 1992.
- [4] Robert Bridson and Matthias Müller-Fisher. *Fluid Simulation.* CRC Press, 2007.
- [5] Ignacio Llamas ByungMoon Kim, Yingjie Liu and Jarek Rossignac. *Advections with Significantly Reduced Dissipation and Diffusion*, volume 31(1). IEEE Transactions on Visualization and Computer Graphics, Georgia Institute of Technology.
- [6] C++ Standards Committee. *ISO/IEC 14882: 2011, Standard for Programming Language C++.* <http://www.open-std.org/jtc1/sc22/wg21>, 2011.
- [7] John B. Bell Elbridge Gerry Puckett, Ann S. Almgren, Daniel L. Marcus, and William J. Rider. *A High-Order Projection Method for Tracking Fluid Interfaces in Variable Density Incompressible Flows.* Journal of Computational Physics, 1997.
- [8] Charles F. Golub, Gene H.; Van Loan. *Matrix Computations.* Johns Hopkins, 1996.
- [9] F. Harlow and J. Welch. *Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface*, volume 8(12). Physics of Fluids, 1965.
- [10] Mark J. Harris. *Fast Fluid Dynamics Simulation on the GPU. In: GPU Gems.* http://http.developer.nvidia.com/GPUGems/gpugems_ch38.html, University of North Carolina at Chapel Hill, 2007.
- [11] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra.* Society for Industrial and Applied Mathematics, 2000.
- [12] Thomas Dornseifer Michael Griebel and Tilman Neunhoeffer. *Numerical Simulation in Fluid Dynamics.* Society for Industrial and Applied Mathematics, USA, 1998.

- [13] Wright S. Nocedal J. *Numerical Optimization*. Springer, 2006.
- [14] Aline Normoyle. *Fluid Simulation Tutorial*. <http://www.alinenormoyle.com/TutorialFluid.html>, University of Pennsylvania, 2011.
- [15] Solarian Programmer. *My programming ramblings*. <https://solarianprogrammer.com/categories/OpenGL>.
- [16] J.A. Sethian. *Level Sets Methods and Fast Marching Methods*. Cambridge University Press, University of California, Berkeley, 1999.
- [17] J.A. Sethian and Peter Smereka. *Level Set Methods for Fluid Interfaces*, volume 35(1). Annual Review of Fluid Mechanics, USA, 2003.
- [18] Jos Stam. *Real-Time Fluid Dynamics for Games*. Proceedings of the Game Developer Conference, Toronto, Ontario, Canada.

Spis rysunków

2.1	Siatka MAC grid w 2D [4]	14
2.2	Komórka siatki MAC grid w 3D [4]	14
2.3	Algorytm adwekcji [10]	16
2.4	Markery cząstek [12]	22
3.1	Wydajność algorytmów. Od lewej: Gaussa-Seidla, Conjugate-Gradient, Conjugate-Gradient prekondycjonowany wynikami z poprzedniego kroku	26
3.2	Algorytm Gaussa-Seidla, liczba iteracji: 100	27
3.3	Algorytm Gaussa-Seidla, liczba iteracji: 300	27
3.4	Algorytm Conjugate-Gradient, liczba iteracji: 20	27
3.5	Algorytm Conjugate-Gradient, liczba iteracji: 80	28
3.6	Ciśnienie wywierane na ściany naczynia	29
3.7	Wizualizacja eksperymentu falującej wody	30
3.8	Wizualizacja eksperymentu wirującej wody, $\Delta t = 0.01s$	31
3.9	Wizualizacja wiru. Od lewej: $\Delta t = 0.01s$, $\Delta t = 0.05s$, $\Delta t = 0.1s$	32
3.10	Woda po powrocie do stanu pierwotnego. Od lewej: $\Delta t = 0.01s$, $\Delta t = 0.05s$, $\Delta t = 0.1s$	32
3.11	Z lewej woda, z prawej ciśnienie dla prędkości stycznej równej $5\frac{m}{s}$	33
3.12	Z lewej woda, z prawej ciśnienie dla prędkości stycznej równej $10\frac{m}{s}$	33

3.13 Z lewej woda, z prawej ciśnienie dla prędkości stycznej równej $20 \frac{m}{s}$	34
3.14 Wizualizacja ruchu wodą przez zwężenie	35
3.15 Wizualizacja rozlewającej się wody	37
3.16 Wizualizacja rozlewającej się wody napotykającej na granicę	38
3.17 Wizualizacja rozlewającej się wody napotykającej na granicę	39

Skorowidz

- algorytm adwekcji, 15, 41
- algorytm projekcji, 18, 42
- algorytm symulacji, 11, 17
- bfecc, 41
- C++, 7
- definicja symulacji komputerowej, 1
- divergence-free, 5
- dyskretyzacja, 11, 18
- dyskretyzacja Eulera, 13
- dyskretyzacja Lagrange'a, 13
- dyssypacja, 17, 32, 38
- eksperymenty, 25
- ekstrapolacja, 17, 37
- eliminacja Gaussa, 20
- falowanie wody, 29
- freetype, 7
- GLFW, 7
- GLM, 7
- Heightfield Approximation, 1
- Incomplete Cholesky, 42
- interpolacja, 15
- level set, 21, 41
- MAC grid, 13, 18
- mapy wysokości, 1
- markery częstek, 21
- metoda Conjugate-Gradient, 20, 25, 26
- metoda Gaussa-Seidla, 20, 25, 26
- metoda gradientu sprzężonego, 20, 25, 26
- metoda Jacobiego, 20
- Navier-Stokes, 1–3, 5, 11
- OpenGL, 7
- pole gęstości, 21
- równanie macierzowe, 20
- równanie Newtona, 3
- równanie ruchu, 2, 3, 11, 18
- ruch wirowy, 30
- signed distance, 21
- twierdzenie Ostrogradskiego-Gaussa, 5
- vorticity confinement, 41
- warunek nieściśliwości, 2, 5, 11, 18
- warunki brzegowe, 7, 11, 18
- wizualizacja, 29, 31, 35, 37
- woda lejąca się z kranu, 36
- woda stojąca, 25
- zwężenie, 35