Nr: 52-IWP-101987-PI-2013

TECHNOLOGIA CUDA I JEJ ZASTOSOWANIE W NUMERYCZNEJ MECHANICE PŁYNÓW

Student: ***Roman GAWRYSIAK***

Promotor: Prof. dr hab. inż. Marek MORZYŃSKI

STRESZCZENIE

W pracy przedstawiono kwestie związane z obliczeniami w technologii Nvidia CUDA . Obejmują one zasadę działania, strukturę, zakres zastosowań oraz zagadnienia dotyczące możliwości sprzętowych kart graficznych z technologią CUDA. Ukazany został proces przeprowadzania obliczeń w wybranych programach i kluczowe pojęcia związane z metodami wykorzystanymi do wykonania symulacji w dziedzinie numerycznej mechaniki płynów. Efektem obliczeń wykonanych zgodnie z opisanymi mechanizmami są symulacje przepływu wokół wybranych geometrii: walca, prostopadłościanu, kuli i ciała Ahmeda.

SUMMARY

The thesis deals with issues related to computations done with the help of Nvidia CUDA technology. It focuses on graphics cards with CUDA support - how they work, their structure and the scope of their applications as well as on their capabilities. The thesis includes a theoretical description of the computational process in selected programs. It also explains key notions related to the methods of making CFD simulations. The thesis presents simulations of flow around different geometries: a cylinder, a cuboid, a sphere and an Ahmed body.

SPIS TREŚCI

1. WSTĘP

1.1. Cel i zakres pracy

Celem pracy jest przedstawienie zastosowania technologii Nvidia CUDA w zakresie obliczeń inżynierskich, a w szczególności obliczeniowej mechaniki płynów oraz przeprowadzenie symulacji w wybranych programach.

Zakres pracy obejmuje teoretyczne omówienie wybranych zagadnień związanych z technologią CUDA: istotę działania, cechy charakterystyczne i zastosowanie. Część praktyczna związana jest z opisem wykorzystanego oprogramowania, metod obliczeniowych i procesem wykonywania symulacji w aplikacjach wykorzystujących CUDA. Efektem przeprowadzonych obliczeń są wizualizacje, np. w programie ParaView.

1.2. Wprowadzenie

Wraz z pojawieniem się pierwszych komputerów, zdefiniowane zostało pojęcie mocy obliczeniowej. Rozumiemy ją jako liczbę działań arytmetycznych, jakie może wykonać jednostka obliczeniowa w danym czasie. Ciągły rozwój informatyki powoduje wzrost zapotrzebowania na większe moce obliczeniowe. Efektem tego popytu, jest stale rosnąca wydajność obliczeniowa komputerów.

Jednostką używaną do określenia tej własności, jest FLOPS[1]. W latach 60 XX wieku, komputery osiągały moc mierzoną w MFLOPS. W roku 1985 „Cray-2" przekroczył pułap 1 GFLOPS, jego następcą został, w 1997 roku „ASCI Red" firmy Intel, o możliwościach powyżej 1 TFLOPS. Kolejnym przełomem był „Roadrunner" wyprodukowany przez IBM w roku 2008 który posiadał moc obliczeniową większą niż 1PFLOPS. Obecny rekord należy do chińskiego „Tiahne-2" i wynosi 33,86 PFLOPS. Tak prężny rozwój informatyki, stawia coraz wyższe wymagania producentom procesorów jak i oprogramowania. Opracowanie nowych technologii, zwiększanie ilości rdzeni CPU[2], oraz wykorzystanie kilku procesorów jednocześnie, wpłynęło na rozwój programowania. Programiści muszą tworzyć aplikacje w taki sposób, aby umożliwiały użycie pełnej mocy procesorów.

[1] Ang. Floating point Operations Per Second - liczba operacji zmiennoprzecinkowych na sekundę
[2] Ang. Central Processing Unit - jednostka przetwarzania centralnego

Rząd wielkości	Oznaczenie	FLOPS
megaflops	MFLOPS	10^6
gigaflops	GFLOPS	10^9
teraflops	TFLOPS	10^{12}
petaflops	PFLOPS	10^{15}

Tab. 1. Wartości operacji zmiennoprzecinkowych na sekundę

W 1999 roku firma Nvidia wprowadziła pierwszy GPU[3], co zapoczątkowało „ruch" GPGPU[4]. Niestety, technologia ta niosła za sobą pewne ograniczenia. Programiści zostali zmuszeni, do zmiany swoich programów, tak by działały jak programy graficzne, a rozwiązania pojawiały się w postaci wielokątów. Znając potencjał obliczeniowy GPU, firma Nvidia rozpoczęła pracę nad modernizacją układu, aby uczynić go w pełni programowalnym i użytecznym w zastosowaniach naukowych, dzięki czemu wprowadziła w roku 2004 technologię CUDA. Pojawienie się takiego rozwiązania, umożliwiło jej zastosowanie w większości tych dziedzin, od przetwarzania wideo, ekonomii, biologii obliczeniowej, aż do symulacji dynamiki płynów.

Technologia ta stworzyła nowe możliwości dla placówek naukowych, ośrodków przemysłowych i użytkowników indywidualnych. Dzięki uzyskaniu ogromnych mocy obliczeniowych, przy niewielkich gabarytach, niższemu zużyciu energii oraz stosunkowo niskiej cenie, można przeprowadzać obliczenia, o niskim lub średnim stopniu skomplikowania, z różnych dziedzin, używając zwykłego komputera PC. Wykorzystanie ogromnych klastrów (połączonych ze sobą GPU), pozwala na rozwiązywanie problemów o szerszej skali, oraz badanie większych i bardziej złożonych systemów.

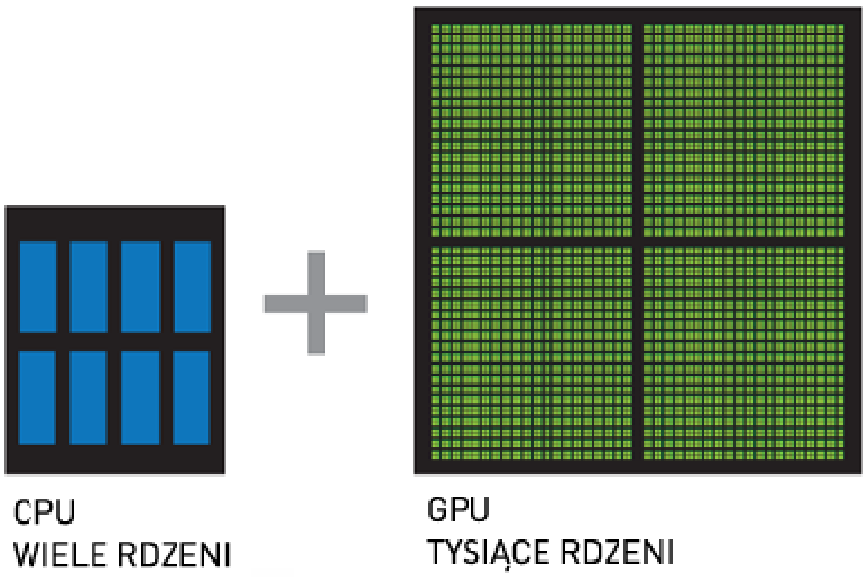
[3] Ang. Graphic Processing Unit – jednostka przetwarzania graficznego
[4] Ang. General-Purpose computation on Graphic Processing Units

2. TECHNOLOGIA CUDA

2.1. Wstęp

CUDA[5], to stworzona przez firmę NVIDIA, równoległa platforma komputerowa i programistyczna, która zezwala na wykorzystanie dużej mocy obliczeniowej procesorów graficznych GPU, w celu rozwiązywania problemów numerycznych. Technologia ta pozwala na dużo wydajniejsze przeprowadzanie obliczeń w sposób równoległy, a nie jak dotychczas, sekwencyjnie, z wykorzystaniem tylko i wyłącznie CPU.

Architektura CUDA zawiera środowisko programistyczne które zostało stworzone w języku programowania C. Zawiera ono własny kompilator (nvcc), debugger (cuda-dbg), który pozwala na sprawdzanie kodu wykonywanego przez CPU i GPU, a także profiler oraz interfejs programowania aplikacji. Do dyspozycji programisty, dostępne są biblioteki umożliwiające wykorzystanie technologii w językach Fortran, Python, Java, Matlab. Kolejnym założeniem CUDA, jest pełna skalowalność programów. Oznacza to, że raz napisana aplikacja, będzie działać na coraz to nowszych procesorach graficznych o większych możliwościach.



Rys. 1. Współpraca CPU i GPU w technologii CUDA [9]

[5] Ang. Compute Unified Device Architecture

2.2. Różnica między CPU a GPU

Aby w pełni zrozumieć na czym polegają obliczenia w technologii CUDA, należy w skrócie przybliżyć czym jest CPU i GPU, oraz główne różnice pomiędzy nimi.

CPU jest to jednostka przetwarzania centralnego. Służy do wykonywania instrukcji nadanej przez program, poprzez wykonywanie podstawowych operacji arytmetycznych, logicznych, oraz operacji wejścia/wyjścia. Wyróżniamy dwa typowe elementy procesora:

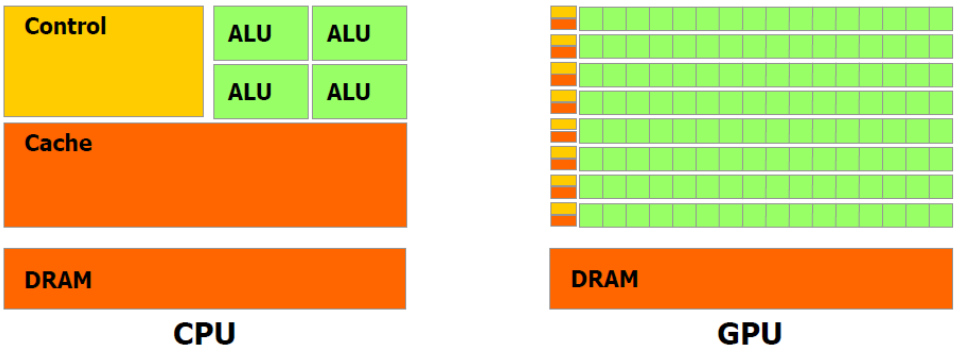
– **ALU**[6] - wykonuje operacje arytmetyczno-logiczne.

– **CU**[7] - steruje operacjami procesora i kontroluje komunikacje i koordynację pomiędzy urządzeniami wejścia i wyjścia. Odczytuje i interpretuje instrukcje, a następnie decyduje kolejność przesyłu danych.

– **FPU**[8] - wspomaga procesor w obliczeniach zmiennoprzecinkowych do których należą np. dodawanie, mnożenie, dzielenie.

– **CACHE** – wbudowana pamięć podręczna działająca podobnie do pamięci RAM, dzięki niej procesor nie oczekuje na dane potrzebne do wykonania zadania.

GPU to jednostka przetwarzania graficznego, początkowo wykorzystywana tylko do renderowania grafiki i przetwarzania jej na sygnał zrozumiały dla urządzenia wyświetlającego. W swojej budowie zawiera od kilku do kilkuset rdzeni więcej niż CPU, co w konsekwencji sprawia, że jest dostosowana do rozwiązywania problemów które da się zrównoleglić – czyli wykonywać te same operacje na dużej ilości informacji.



Rys. 2. Porównanie architektury CPU I GPU [10]

[6] Ang. Arithmetic and Logic Unit
[7] Ang. Control Unit
[8] Ang. Floating Point Unit.

2.3. Obliczenia równoległe

Kolejnym bardzo ważnym aspektem związanym z technologią Nvidia CUDA, są obliczenia równoległe. Programowanie w taki sposób pozwala na wykorzystanie mocy obliczeniowej wielordzeniowych kart graficznych Nvidia do rozwiązywania problemów z różnych dziedzin. Aby w zrozumiały sposób przedstawić istotę obliczeń równoległych, należy zacząć od opisania podstawowych pojęć.

Pierwszym zagadnieniem jest **program sekwencyjny**. Jest to pewien ciąg instrukcji który ma rozwiązać postawiony problem, czyli wykorzystać dane wejściowe dla uzyskania danych wyjściowych (wyniku). Przyjęto, że taki program jest wykonywany przez jeden procesor.

Drugim będzie zatem **program równoległy**. Jest to program gdzie problem główny zostaje podzielony na podproblemy, mają one charakter programu sekwencyjnego, dla uproszczenia można nazywać je zadaniami (Ang. task), a następnie przydziela je do rozwiązania, dostępnym procesorom . Należy pamiętać, że zadania muszą być możliwe do rozwiązania w sposób równoległy.

Zasadniczą różnicę pomiędzy powyższymi programami, można zobrazować na przykładzie dodawania dwóch macierzy. Program sekwencyjny rozwiąże problem korzystając z jednego procesora w danym momencie, co oznacza, że wykona operacje dodawania kolejnych składowych macierzy. Dla przykładu:

$$\begin{bmatrix} 1{,}3 & 2 & 3 \\ 1 & 2 & 9 \end{bmatrix} + \begin{bmatrix} 1{,}2 & 2 & 11 \\ 3 & -4 & 7 \end{bmatrix} = \begin{bmatrix} 2{,}5 & 4 & 14 \\ 4 & -2 & 16 \end{bmatrix}$$

Program sekwencyjny wykona działania po kolei, aż do uzyskania wyniku końcowego. Program równoległy podzieli problem główny (dodawanie całej macierzy) na mniejsze zadania (dodawanie składowych macierzy), następnie dokona przypisania zadań do poszczególnych procesorów. Po wydaniu instrukcji nastąpi jednoczesne rozwiązanie problemów składowych w pojedynczych procesorach.

Bardziej szczegółowy opis zaprezentowanego zagadnienia można odnaleźć w literaturze specjalistycznej[9].

9

Zwiększenie wydajności uzyskanej przez aplikacje korzystające z technologii CUDA, zależy całkowicie od tego, w jakim stopniu może zostać zrównolegniona. Prawo Amdahl'a daje nam możliwość oszacowania maksymalnego wzrostu wydajności, spowodowanej paralelizacją fragmentów kodu. Zgodnie z zasadą, maksymalne przyspieszenie wynosi:

$$S = \frac{1}{(1 - P) + \frac{P}{N}} \qquad (1)$$

Gdzie:

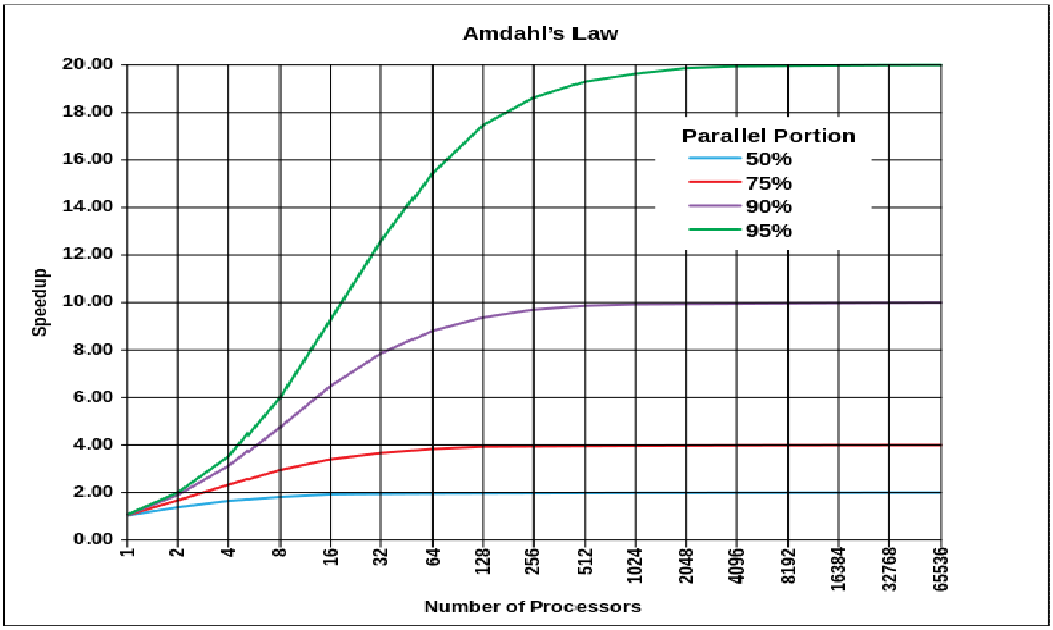
P - porcja kodu, którą można sparalelizować

N - ilość procesorów na których liczona jest partia równoległego kodu

Analizując powyższe równanie, im więcej procesorów zastosujemy, tym mniejszy będzie ułamek P/N. Jeśli przyjmiemy, że liczba procesorów jest bardzo duża, równanie sprowadza się do postaci:

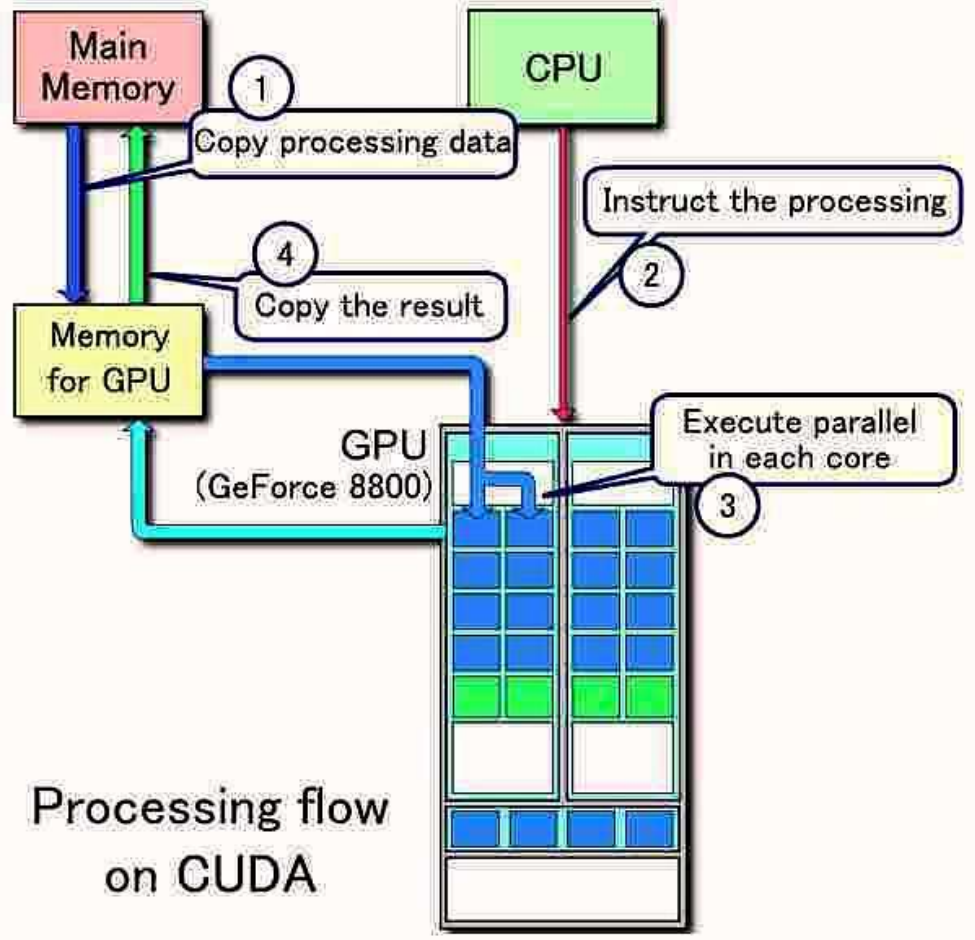
$$S = \frac{1}{(1 - P)} \qquad (2)$$

Oznacza to, że jeśli ¾ programu jest napisane w sposób równoległy, to osiąga się 4-krotnie zwiększenie szybkości działania. Drugim oczywistym wnioskiem jest fakt, iż należy skupić się na jak największym zrównolegleniu kodu w celu uzyskania wzrostu wydajności



Rys. 3. Wpływ ilości procesorów i paralelizacji kodu na zwiększenie wydajności CPU I GPU [17]

2.4. Strumień przetwarzania w technologii CUDA



Rys. 4. Przepływ informacji w technologii CUDA [9]

Ad 1. Przesył danych z pamięci głównej do GPU.

Ad 2. CPU wysyła instrukcje odnośnie procesu do GPU.

Ad 3. GPU wykonuje instrukcje równolegle na wszystkich rdzeniach.

Ad 4. Przesył wyników z GPU do pamięci głównej.

2.5. Struktura CUDA

Do wykorzystania technologii CUDA, potrzebne są następujące elementy, składające się na jej indywidualną strukturę:

– odpowiednio zaprogramowana aplikacja

– biblioteki CUDA:

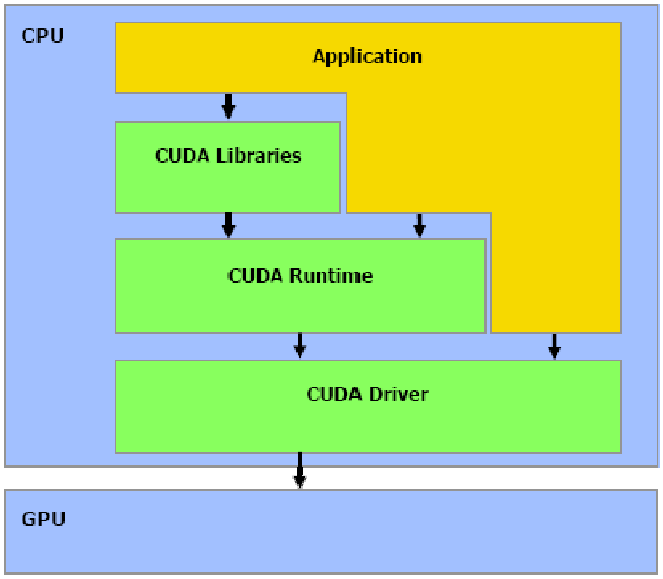
a) cuFFT[10] - zoptymalizowana biblioteka, zawierająca algorytm, do wykonywania transformacji Fouriera dla zestawów danych. Używany głównie w fizyce obliczeniowej i przesyle sygnałów.

b)cuBLAS[11] - rozwiązuje podstawowe problemy algebry liniowej

– CUDA runtime - API[12] wyższego poziomu

– CUDA driver - API niższego poziomu

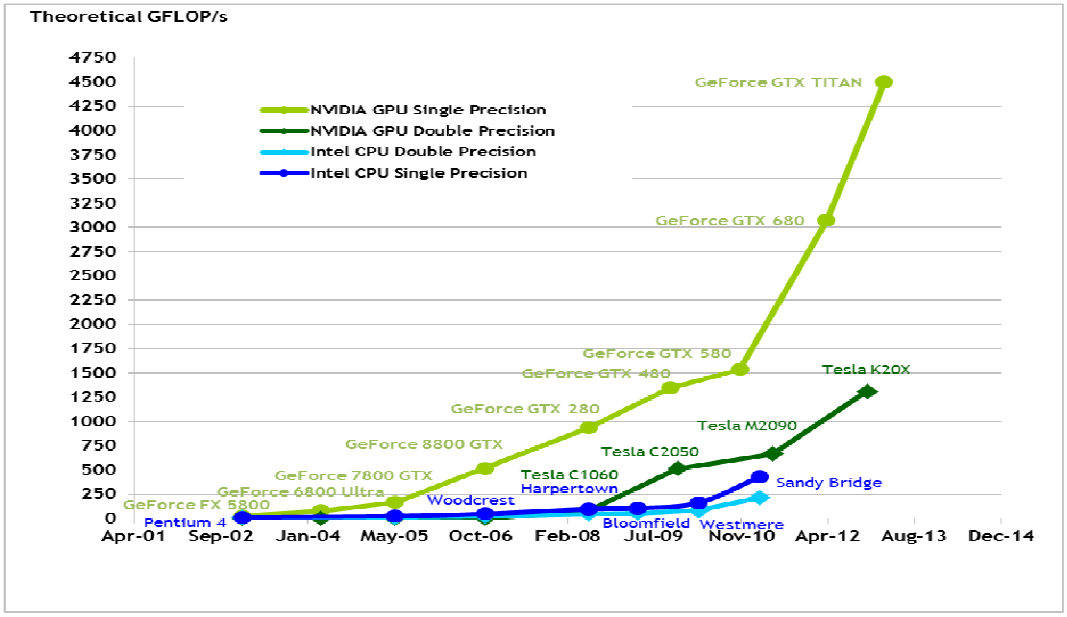
– GPU – procesor graficzny wykorzystujący technologię CUDA



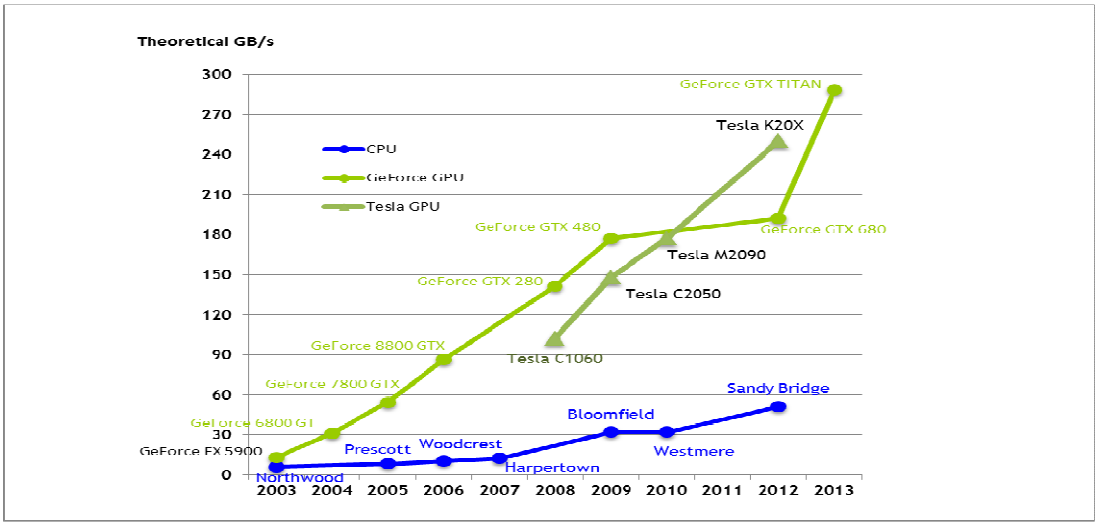
Rys. 5. Struktura CUDA [10]

[10] Ang. Fast Fourier Transform
[11] Ang. Basic Linear Algebra Subprograms
[12] Ang. Application Programming Interface – zestaw reguł w jaki program komunikują się między sobą

2.6. Porównanie wydajności CPU I GPU



Rys. 6. Porównanie możliwości wykonywania operacji zmiennoprzecinkowych CPU i GPU [9]



Rys. 7. Porównanie przepustowości danych CPU i GPU [9]

W przeciągu kilkunastu lat nastąpił znaczący rozwój technologii Nvidia CUDA. Podczas tak krótkiego okresu czasu, znacząco powiększyła się przepaść pomiędzy wydajnością obliczeniową procesorów graficznych i centralnych jednostek przetwarzania. Szybki rozwój spowodowany jest stale rosnącym zapotrzebowaniem, we wszystkich dziedzinach nauki, na wysokowydajne aplikacje obliczeniowe.

2.7. Karty graficzne Nvidia CUDA

Niezbędnym elementem architektury CUDA jest część związana bezpośrednio z hardware, czyli karta graficzna. Poniżej przedstawiono wybrane procesory GPU wraz ze specyfikacją[15]:

– GeForce GTX TITAN:

a) liczba rdzeni CUDA: 2688
b) częstotliwość bazowa: 837MHz
c) szybkość pamięci: 6.0 Gbps
d) pojemność pamięci: 6144 MB
e) interfejs pamięci 384 – bit GDDR5

Rys. 8. GeForce GTX TITAN [15]

– Quadro K6000 (karta dedykowana do obliczeń numerycznych):

a) liczba rdzeni CUDA: 2880
b) gigaflopy (pojedyncza precyzja): 1030,4
c) gigaflopy (podwójna precyzja): 515,2
d) szybkość pamięci: 6.0 Gbps
e) pojemność pamięci: 12GB DDR5
f) interfejs pamięci 384 – bit GDDR5

Rys. 9. Nvidia Quadro K6000 [15]

– Tesla K20 (karta dedykowana do obliczeń numerycznych):

a) liczba rdzeni CUDA: 2496
b) szczytowa wartość FLOPS o podwójnej prezycji: 1,17 TFlops
c) szczytowa wartość FLOPS o pojedynczej precyzji: 3,52 TFlops
d) przepustowość: 208GB/s Gbps
e) pojemność pamięci: 5GB DDR5
f) interfejs pamięci 384 – bit GDDR5

Rys. 10. Tesla K20 [15]

– Tesla K20 (karta dedykowana do obliczeń numerycznych):

a) liczba rdzeni CUDA: 2880
b) szczytowa wartość FLOPS o podwójnej prezycji: 1,43 TFlops
c) szczytowa wartość FLOPS o pojedynczej precyzji: 4,29 TFlops
d) przepustowość: 288GB/s Gbps
e) pojemność pamięci: 12GB DDR5

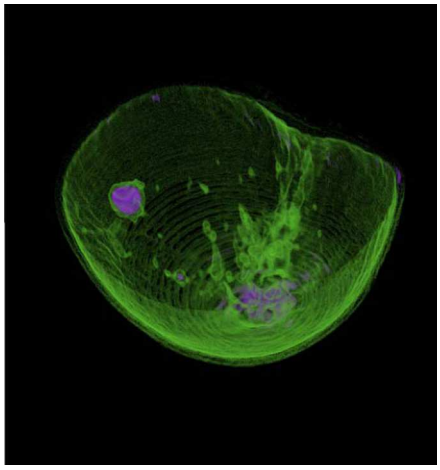
Rys. 11. Tesla K20 [15]

2.8. Zastosowania CUDA

Od momentu debiutu technologii CUDA w 2007 roku, wiele sektorów przemysłu na świecie, odniosło znaczący sukces tworząc swoje programy w języku CUDA C. Dzięki zastosowaniu nowatorskiego podejścia do problematyki obliczeniowej, uzyskano znaczny wzrost wydajności aplikacji przy jednoczesnym obniżeniu poboru mocy przez jednostki robocze. Wdrożono także wiele rozwiązań, które do tej pory nie były możliwe, z powodu ograniczeń sprzętowych. Oto kilka zastosowań technologii CUDA:

– Obrazowanie medyczne

Głównym problemem przy profilaktyce tak powszechnego schorzenia, jakim jest rak piersi, jest jego diagnozowanie. Szeroko stosowany mammograf ma kilka ograniczeń. Należy stworzyć kilka obrazów, następnie zdjęcia muszą zostać wywołane i przeanalizowane przez lekarza specjalistę, aby wykryć ewentualne guzy. Ten proces może zostać powtórzony kilka razy w celu uzyskania dokładniejszych danych. Pociąga to za sobą konsekwencje w postaci wydłużenia czasu od diagnozy do podjęcia odpowiednich kroków mających na celu wyleczenie pacjenta oraz narażenie badanego na kolejne dawki promieniowania rentgena skierowane na jego klatkę piersiową. Używanie ultrasonografu jest bezpieczniejsze, ale pociąga za sobą jeszcze większe ograniczenia. Dzięki obliczeniom na GPU w technologii CUDA został opracowany specjalny ultrasonograf o nazwie TechniScan. Oparty na dwóch kartach TESLA, przetwarza 35GB danych stworzonych podczas bezpiecznego 15- minutowego skanowania. Po około 20 minutach lekarz otrzymuje szczegółowy obraz 3D badanej piersi.

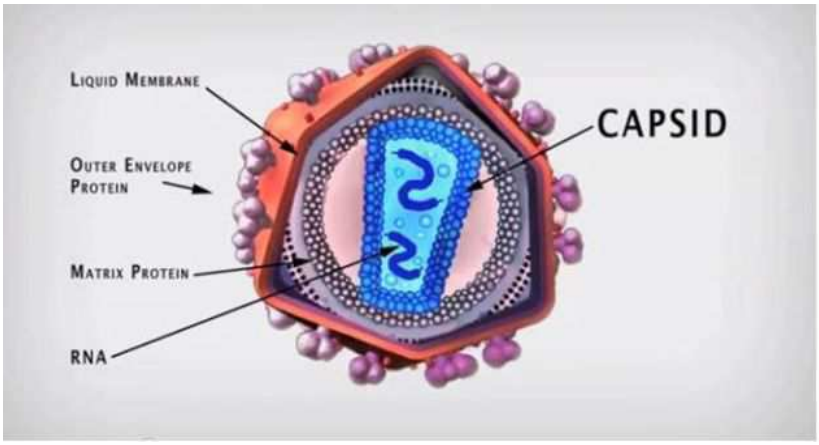


Rys. 12. Przykładowy obraz USG [18]

Rys. 13. USG TechnniScan [18]

– Biologia molekularna

Po raz pierwszy udało się pozyskać precyzyjne informacje odnośnie chemicznej struktury kapsydu[13] zawierającej informacje genetyczne dotyczące wirusa HIV. Jej głównym zadaniem jest ochrona i przenoszenie materiału genetycznego wirusa do organizmu człowieka. Symulacja została przeprowadzona na superkomputerze „Blue Waters", wyposażonym w 3000 akceleratorów graficznych Nvidia Tesla K20X. Obejmowała ona aż 64 miliony molekuł. Wyniki tych badań w znacznym stopniu wpłyną na opracowanie skutecznych leków przeciwko temu wirusowi.



Rys. 14. Budowa wirusa HIV [19]

Pozostałe dziedziny w których stosuje się technologię CUDA:

– obliczenia finansowe

– materiałoznawstwo

– analiza numeryczna

– fizyka

– chemia Kwantowa

– badania sejsmiczne

– prognozowanie pogody i klimatu

– obronność i Inteligencja

– numeryczna mechanika płynów

Szczegółowy opis zastosowań w wyżej wymienionych dziedzinach można znaleźć na stronie głównej Nvidia.

[13] Kapsyd – osłona wirusa, złożona z białka

3. OBLICZENIA Z UŻYCIEM TECHNOLOGII CUDA

3.1. DualSPHysics

Projekt Open Source[14] wykorzystujący metodę SPH[15] do wykonywania symulacji z dziedziny mechaniki płynów. Kod jest połączeniem trzech języków programowania C++, CUDA i Java. Aplikacja ta produktem współpracy naukowców z trzech uczelni: Johns Hopkins University (Stany Zjednoczone), University of Vigo (Hiszpania) oraz University of Manchester (Wielka Brytania).

Rys. 15. logo programu [13]

W wersji obecnej program stosowany jest głównie do rozwiązywania następujących zagadnień:

– problemy hydrauliczne
– zastosowania przemysłowe
– interakcje pomiędzy cieczą a ciałem (możliwość importu skomplikowanych geometrii)
– problem „rozlewania”
– ochrona wybrzeża (np. projektowanie falochronów)
– realistyczne wizualizacje o wysokiej rozdzielczości i dużej ilości cząstek (ponad 1000 mln)

[14] Ang. open source – wolne oprogramowanie
[15] Ang. SPH – smoothed particie hydrodynamics

3.1.1. Metoda SPH

Jedna z kilku metod bezsiatkowych, stosowanych do symulacji zachowania się płynu w CFD[16]. Znajduje zastosowanie w obliczeniach przepływu w których następuje duża deformacja płynu na przykład w astrofizyce, balistyce czy inżynierii przybrzeżnej. Z powodzeniem stosowana jest także w symulacji odlewania wtryskowego.

Płyn zostaje określony jako zbiór cząstek, a każda z nich jest opisana przez kilka wartości[]:

–masa (m_i)

– prędkość (V_i)

– pozycja (r_i)

– siła (F_i)

– gęstość (ρ_i)

– ciśnienie (p_i)

– kolor (C_i)

Przyspieszenie całkowite cząstki definiuje się jako sumę przyspieszeń składowych:

$$\frac{dv_i}{dt} = a_{ip} + a_{iV} + a_{iI} + a_{ig} \tag{3}$$

Gdzie:

a_{ip} – przyspieszenie zależne od ciśnienia

a_{iV} – przyspieszenie zależne od lepkości

a_{iI} – przyspieszenie zależne od interakcji cząstek

a_{ig} – przyspieszenie zależne od siły grawitacji

[16] Ang. CFD – Computational Fluid Dynamics

Masa cząstki:

$$m = \frac{\rho V}{P} \tag{4}$$

Gdzie:

ρ - gęstość całkowita płynu

V - objętość całkowita płynu

P - ilość cząstek w układzie

Gęstość:

$$\rho_i = \Sigma m_j W(r_{ij}) \tag{5}$$

Gdzie:

$W(r_{ij})$ – funkcja wagowa zależna od odległości sąsiadującej cząsteczki

Ciśnienie:

$$p_i = c_s^{\,2}(\rho_i - \rho_0) \tag{6}$$

Gdzie:

c_s – prędkość dźwięku

ρ_0 - gęstość odniesienia płynu

W metodzie wygładzonych cząstek uśrednia się wartości w następujący sposób:

$$< A >_i \approx \Sigma \frac{m_j}{\rho_j} A_j W(r_{ij}) \tag{7}$$

$$< \nabla A >_i \approx \Sigma \frac{m_j}{\rho_j} A_j \nabla W(r_{ij}) \tag{8}$$

$$< \nabla^2 A >_i \approx \Sigma \frac{m_j}{\rho_j} A_j \nabla^2 W(r_{ij}) \tag{9}$$

Metodę uśredniania stosuję się w równaniu Naviera-Stokes'a, które w tym przypadku ma postać:

$$\frac{dv}{dt} = -\frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla * \nabla v + \frac{f_{zew}}{m} + g \tag{10}$$

Ciśnienie:

$$\left\langle -\frac{1}{\rho}\nabla p \right\rangle_i \approx \sum_j P_{ij} \nabla W(r_{ij}) \tag{11}$$

Gdzie:

$$P_{ij} = -m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \tag{12}$$

Lepkość:

$$\left\langle \frac{\mu}{\rho}\nabla * \nabla v \right\rangle_i \approx \sum_j V_{ij} \nabla^2 W(r_{ij}) \tag{13}$$

Gdzie:

$$V_{ij} = \mu m_j \left(\frac{v_j - v_i}{\rho_i \rho_j} \right) \tag{14}$$

Podsumowując, przyspieszenie całkowite przybiera postać:

$$\frac{dv_i}{dt} = a_{ip} + a_{iV} + a_{iI} + a_{ig} \tag{3}$$

$$a_{ip} \approx \sum_j P_{ij} \nabla W(r_{ij}) \tag{15}$$

$$a_{iv} \approx \sum_j V_{ij} \nabla^2 W(r_{ij}) \tag{16}$$

$$a_{iv} \approx \frac{1}{m_i} f_I \tag{17}$$

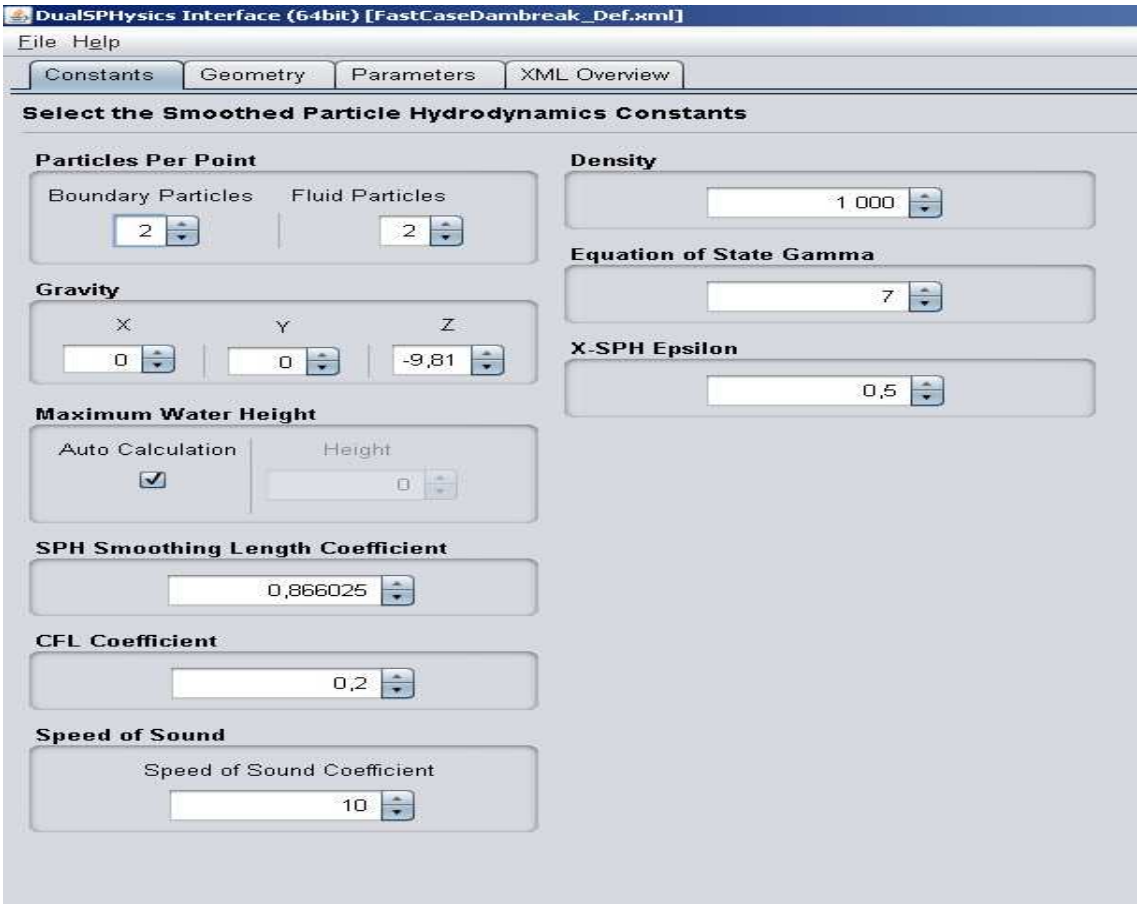
$$a_{ig} \approx (0,0,-g) \tag{18}$$

3.2. Definiowanie symulacji (Pre-processing)

Każda symulacja którą dokonuje się w dziedzinie numerycznej mechaniki płynów, składa się z kilku elementów. Pierwszym z nich jest przygotowanie domeny i określenie warunków symulacji. W przypadku DualSPHysics, można to rozwiązać na dwa sposoby:

a) Stworzenie pliku z danymi w formacie XML, który następnie zostanie przetworzony na kod binarny
b) Korzystanie z interfejsu w języku Java gdzie poszczególne ustawienia zostaną automatycznie umieszczone w pliku XML

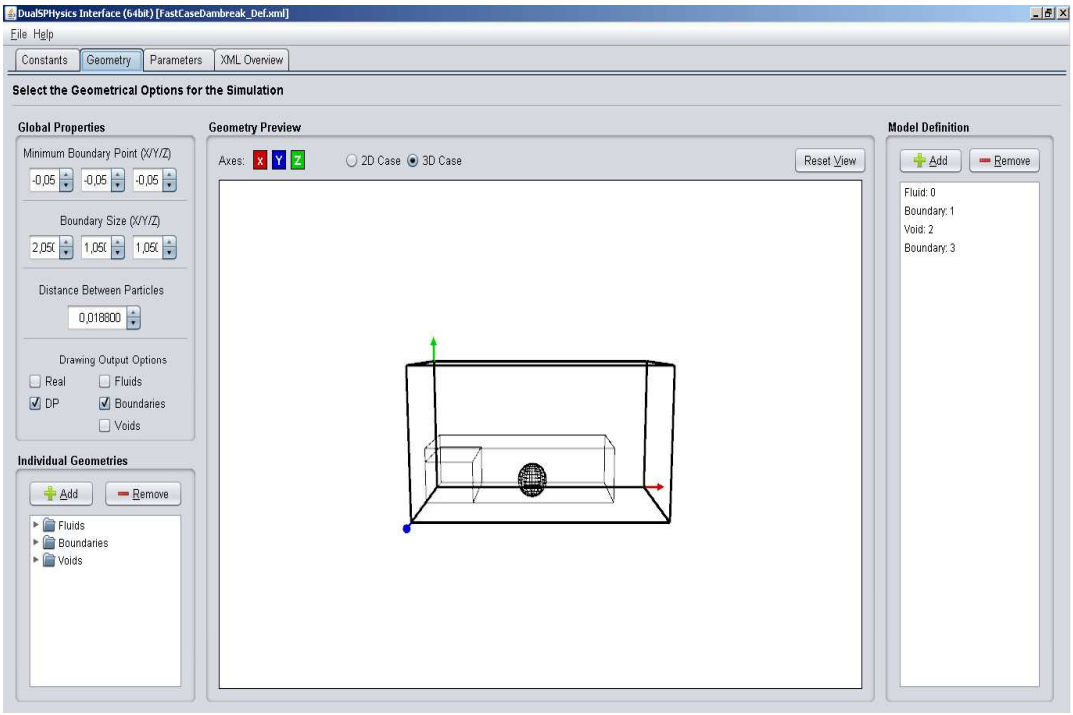
W poszczególnych zakładkach należy zadać parametry dla danego przypadku:



Rys. 16. Zakładka „Constraints”

W pierwszej zakładce zadawane są warunki panujące w układzie np.:

c) ilość cząstek na punkt
d) gęstość pojedyńczej cząstki
e) wartość grawitacji oddziałującej na układ
f) maksymalna wysokość cieczy



Rys.17. Zakładka „Geometry”

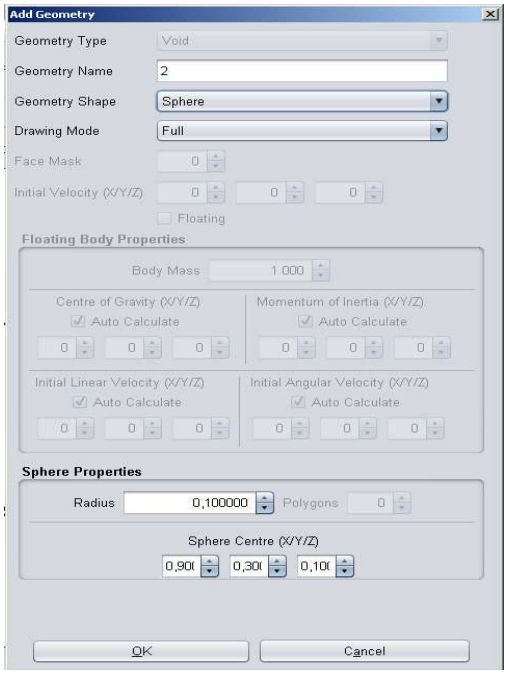
W kolejnej części tworzy się rozpatrywany układ, poprzez :

a) ustalenie wymiarów globalnych i ustalanie geometrii (domeny)

b) rozmieszczenie i określenie współpracy pomiędzy wybranymi geometriami i płynem

W tym przypadku nie było wymagane wykorzystanie dodatkowego oprogramowania do wykonania domeny. Dodatkowe opcje geometrii umożliwiają stworzenie podstawowych figur lub brył geometrycznych oraz określenie ich podstawowych właściwości. Istnieje możliwość nadania jednego z kilku rodzajów ruchu, jakim porusza się dany element, jednak w rozpatrywanych przykładach, nie zostało to wykorzystane.

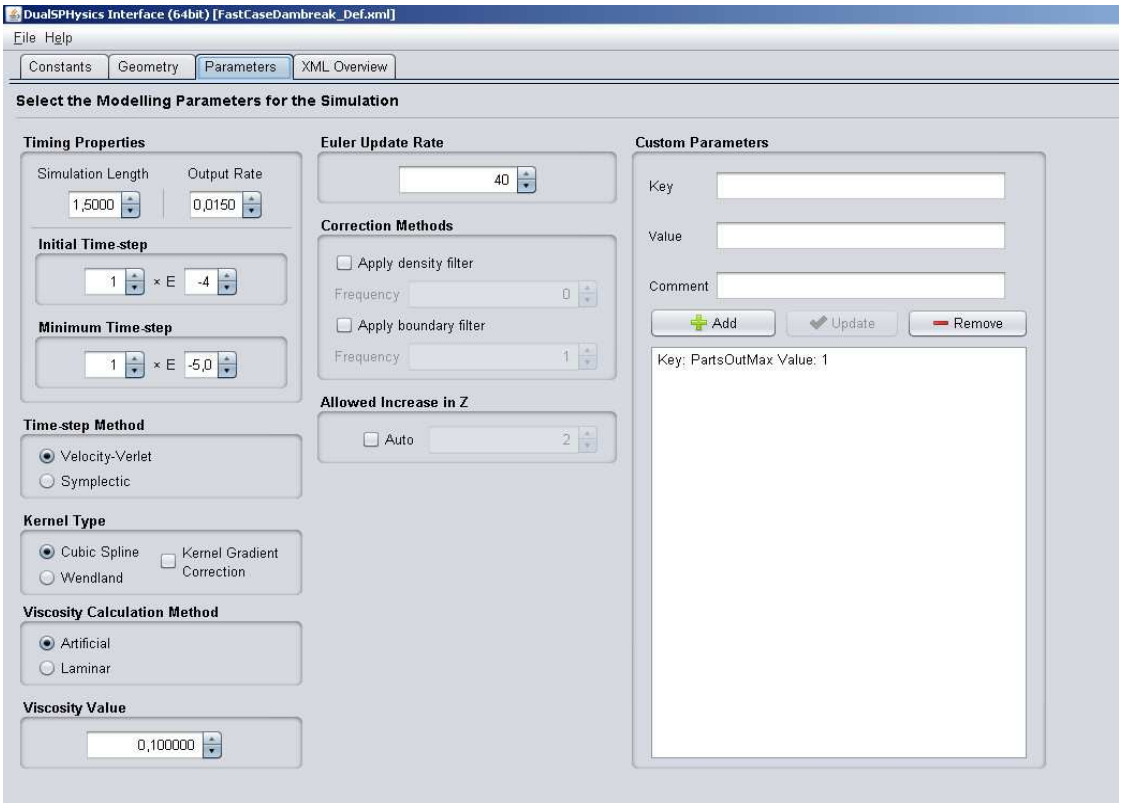
Istnieje możliwość wczytania, bardziej skomplikowanej geometrii, jest to możliwe poprzez import w formacie STEP[17] lub IGES[18].



Rys. 18. Zakładka „ Add Geometry”

[17] STEP - STandardized Exchange of Product

[18] IGES - Initial Graphics Exchange Specification



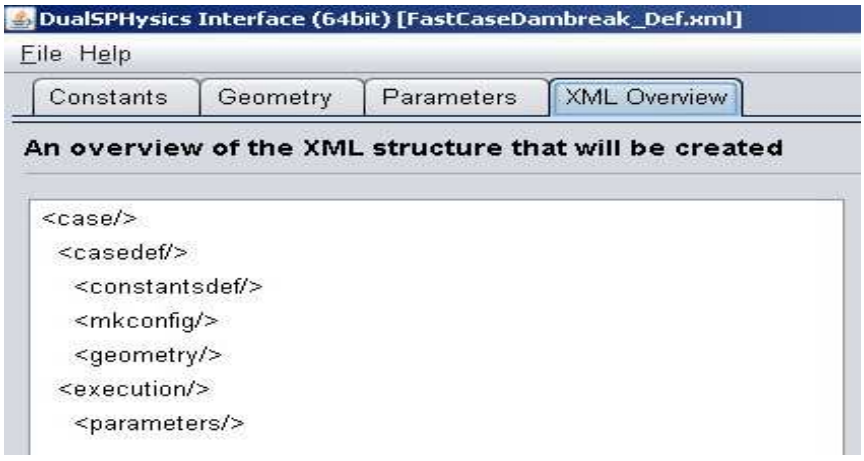
Rys. 19. Zakładka „Parameters”

W tym miejscu ustalone zostają ogólne parametry przeprowadzania symulacji np.:

a) długość symulacji

b) krok czasowy

c) metodę obliczania lepkości płynu



Rys. 20. Zakładka „XML Overview”

W zakładce XML Overwiew ukazana zostaje struktura, która zostanie wytworzona po skonfigurowaniu i zapisaniu symulacji.

3.3. Przeprowadzenie symulacji (Solving)

Po uruchomieniu pliku wykonawczego następuje, przetwarzanie wprowadzonych wcześniej danych. Symulacje w programie „DualSPHysics", zgodnie z metodą SPH, można podzielić na trzy zasadnicze etapy:

Etap I – lista sąsiadów:

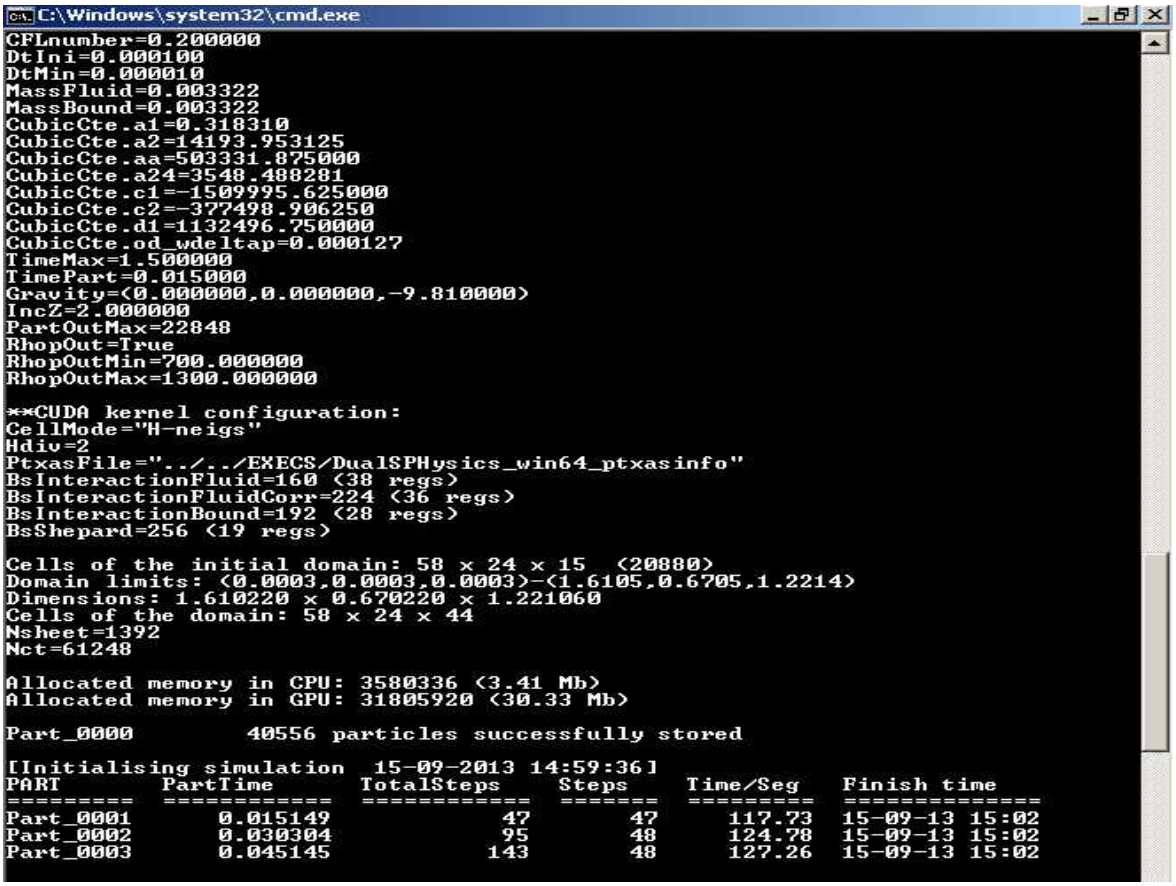
a) domena zostaje podzielona na komórki
b) dla każdej komórki stworzona zostaje lista, znajdujących się w niej cząstek
c) następuje przydzielenie wartości fizycznych dla każdej cząstki

Etap II – obliczanie

a) cząsteczki z tych samych, oraz przylegających komórek, mogą zostać sąsiadami
b) oddziaływanie między sąsiadami (przeliczanie parametrów)

Etap III – Aktualizacja

a) wielkości fizyczne dla każdej cząsteczki zostają uaktualnione i na ich podstawie liczony jest kolejny krok czasowy



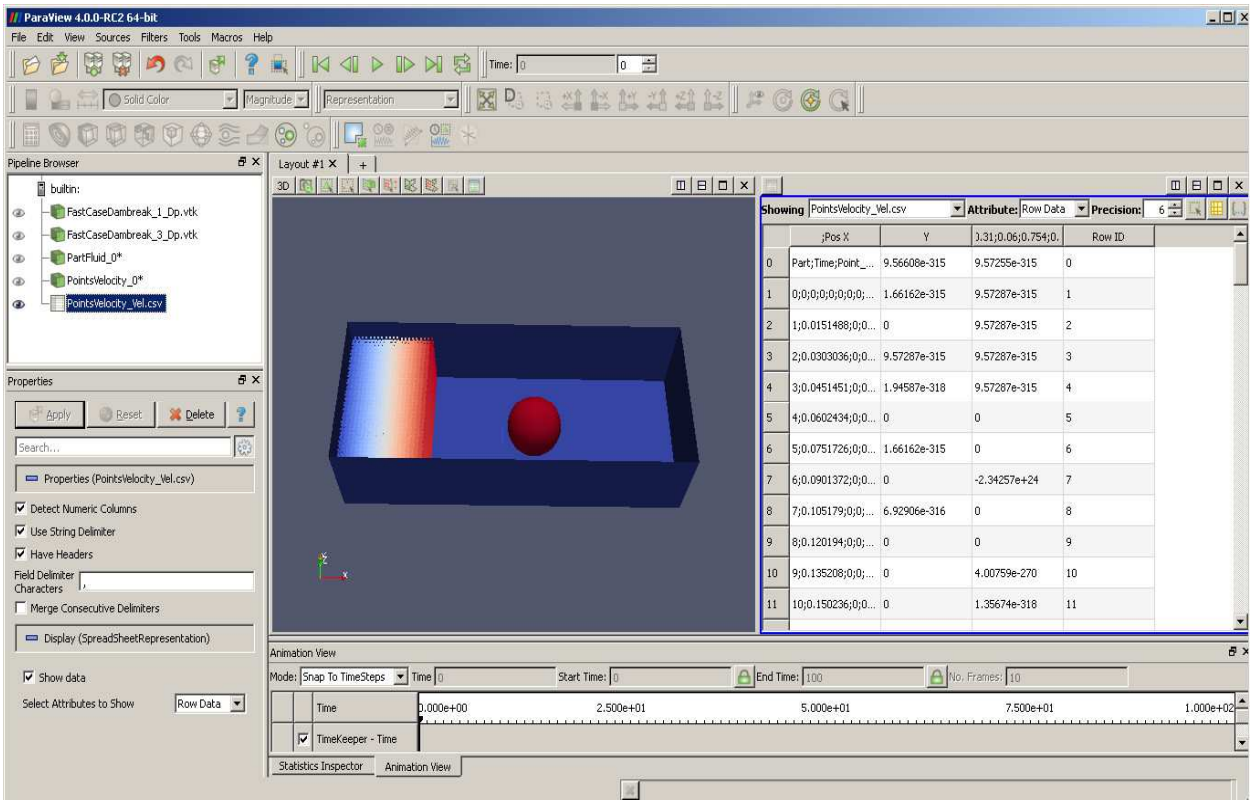
Rys. 21. Okno „Solvera"

3.4. Wizualizacja (Post-processing)

Kolejnym i zarazem ostatnim krokiem symulacji jest przedstawienie wyników obliczeń dokonanych wcześniej przez solver. Wykorzystano do tego program „ParaView", który jest programem typu „OpenSource". Po skończeniu iteracji stworzony zostaje katalog o nazwie symulacji z dopiskiem „out". Znajdują się tam wszystkie pliki potrzebne do wizualizacji efektów wykonanych obliczeń:

a) pliki o rozszerzeniu VTK, zawierające geometrię domeny i wszystkich elementów zawartych w symulacji

b) pliki w których zawarte są parametry cząstek dla każdego kroku czasowego o rozszerzeniu VTK oraz w postaci binarnej



Rys. 22. Okno robocze programu „ParaView"

Poniżej przedstawiono wyniki symulacji interakcji cieczy z powierzchniami swobodnymi w postaci:

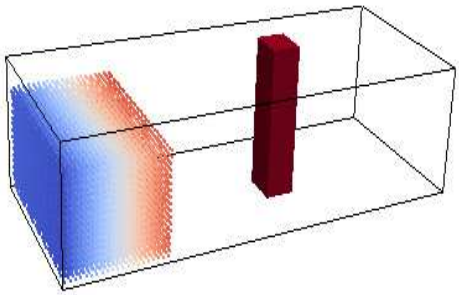
a) prostopadłościanu

b) walca

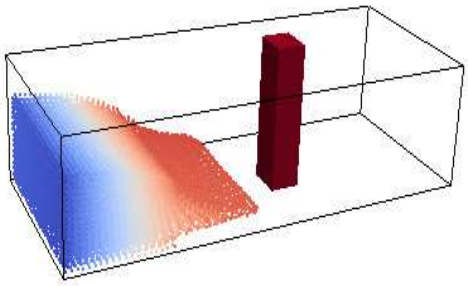
c) kuli

Prostopadłościan

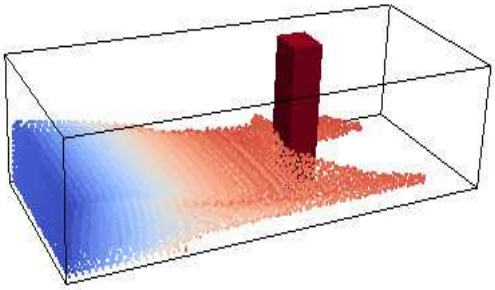
t = 0



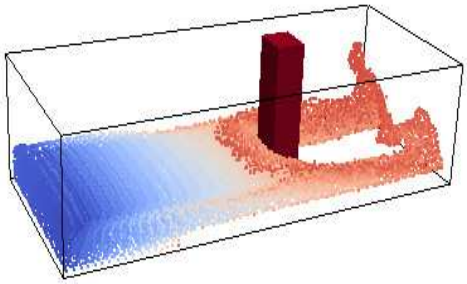
t = 15



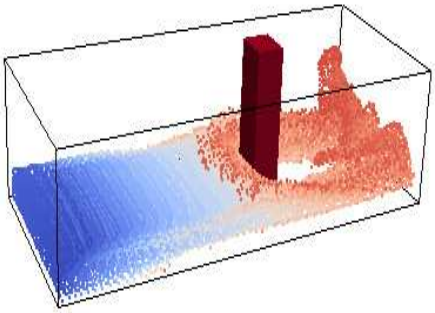
t = 30



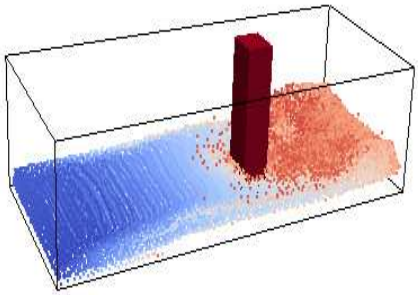
t = 45



t = 60

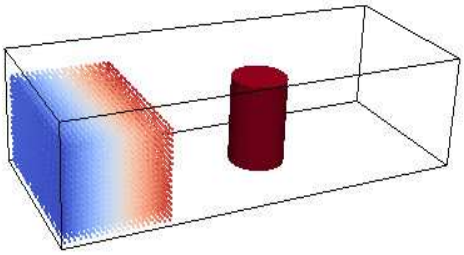


t = 90

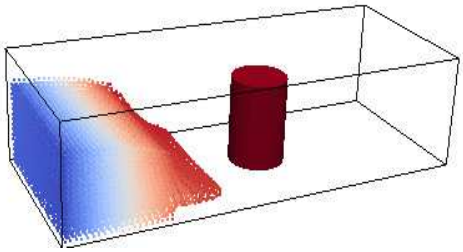


Walec

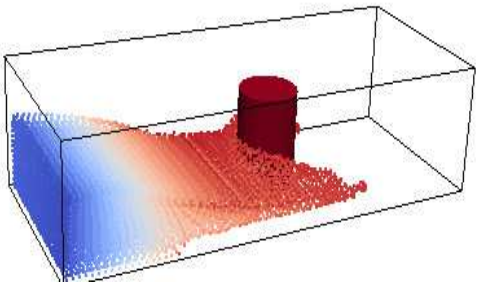
t = 0



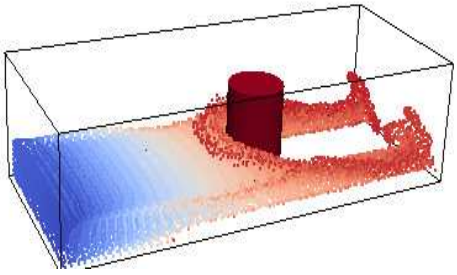
t = 15



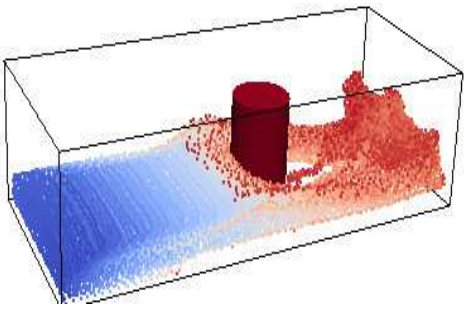
t = 30



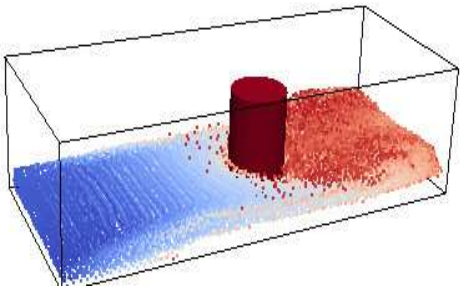
t = 45



t = 60

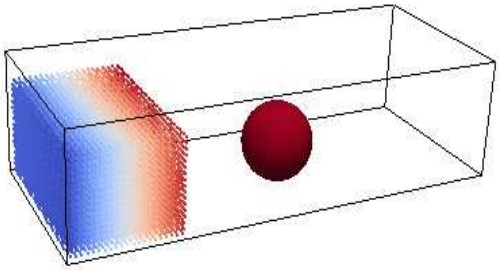


t = 90

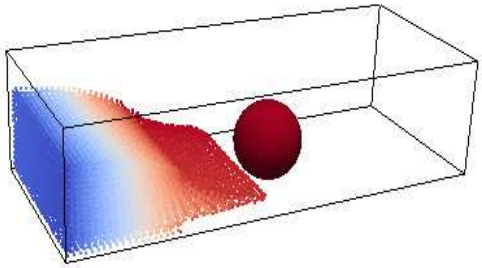


Kula

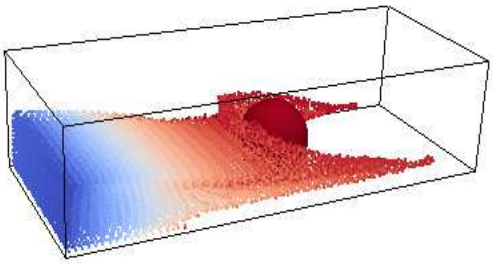
t= 0



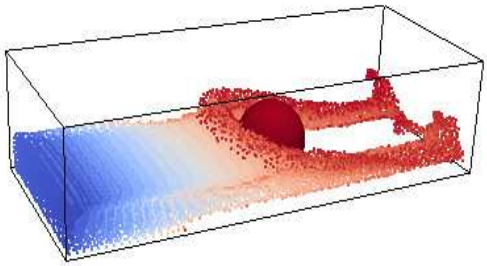
t= 15



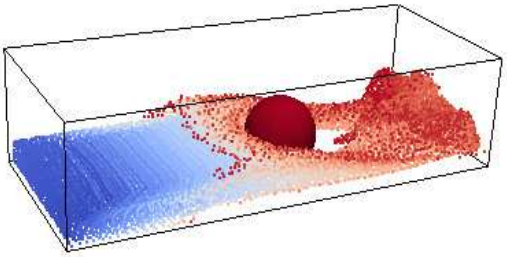
t= 30



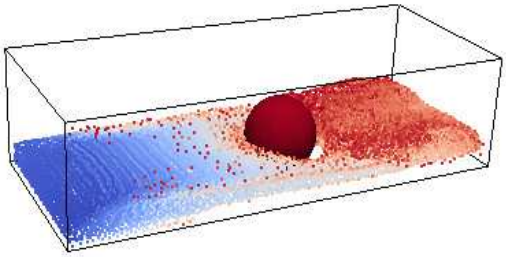
t= 45



t= 60



t= 90



Różnice barw oznaczają zmiany prędkości cząstek w trakcie przepływu. Obliczenia wykonane przy pomocy tego programu zostały wykonane, w komputerze mobilnym z procesorem Intel i3 i kartą graficzną GeForce GT 520M, CUDA COMPUTE: 2.1, (48 rdzeni CUDA).

3.5. Caedium

Drugą aplikacją wykorzystaną do obliczeń w technologii Nvidia CUDA jest Caedium. Aplikacja ta wykorzystuję skrypt OpenFOAM z nakładką ofgpu do przeprowadzania obliczeń metodą RANS[19] i PANEL (metody siatkowe). Przepływ RANS, symuluje realistyczny (lepki) płyn, natomiast PANEL płyn wyidealizowany. Wykonywanie obliczeń ułatwia interfejs graficzny, dzięki któremu można definiować symulacje. W skład oprogramowania wchodzi:

a) System CAD - budowa lub import geometrii
b) Pre-preocessor - tworzenie siatki potrzebnej do obliczeń
c) Solver - konfiguracja i uruchomienie symulacji
d) Post-processor - obrazowanie wyników

Do wykonania obliczeń wykorzystana została darmowa 30-dniowa wersja programu Caedium.

3.6. RANS

Problemy przepływowe opisane są równaniem Naviera-Stokes'a:

$$\frac{\partial v_i}{\partial t} + \frac{\partial v_i v_j}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu + \frac{\partial^2 v_i}{\partial x_j \partial x_j} \qquad (19)$$

Klasyczne modelowanie turbulencji oparte jest na koncepcji Reynoldsa, zgodnie z którą każda wielkość U opisująca przepływ turbulentny może być traktowana jako suma wielkości uśrednionej w czasie $\underline{U}$ oraz składowej fluktuacyjnej u , która to wielkość jest losową funkcją czasu i przestrzeni. Zastosowanie tej koncepcji do równań Naviera – Stokesa (N-S) przekształca je do postaci znanej jako równanie Reynoldsa, które dla przepływu nieściśliwego zapisane być może w postaci[20]:

$$\rho\left(\frac{\partial \overline{U}_i}{\partial t} + \overline{U}_j \frac{\partial \overline{U}_i}{\partial x_j}\right) = \frac{\partial}{\partial x_j}(\sigma_{ij}) + \overline{F}_i \qquad (20)$$

[19] RANS -

Gdzie tensor naprężeń:

$$\sigma_{ij} = -\overline{p}\delta_{ij} + \nu\rho\left(\frac{\partial \overline{U}_i}{\partial x_i} + \overline{U}_j \frac{\partial \overline{U} j}{\partial x_j}\right) - \rho\overline{u_i u_j} \tag{21}$$

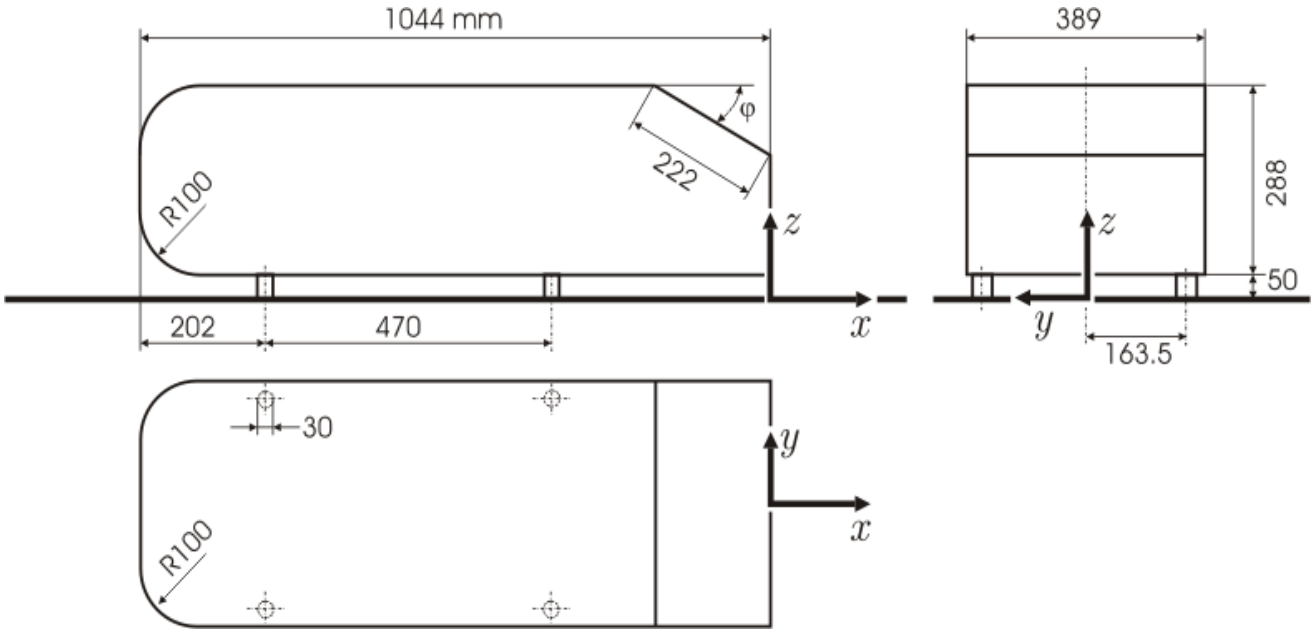
posiada człon który nie występuje w równaniu N-S, czyli tensor naprężeń Reynoldsa:

$$(\sigma_T)_{ij} = -\rho\overline{u_i u_j} \tag{22}$$

pojawienie się dodatkowej wielkości w równaniu N-S, niesie za sobą konsekwencje w postaci niezamkniętego układu równań N-S, w celu jego domknięcia zostają użyte modele turbulencji (hipotezy zamykające). Poszerzony opis metody można znaleźć w literaturze poświęconej numerycznej mechanice płynów.

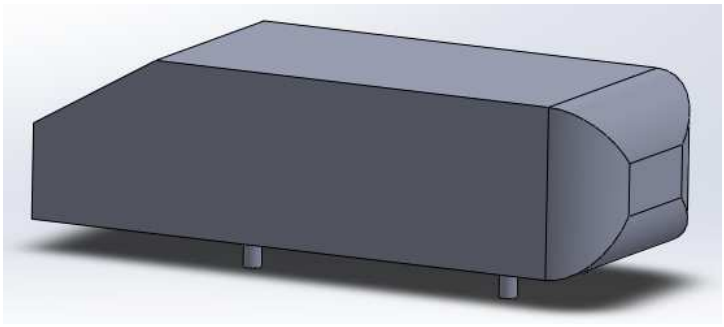
3.7. Przykład własny

Do przeprowadzenia symulacji wybrano opływ powietrza wokół ciała Ahmeda. Jest to często wykorzystywany model do obliczeń przepływowych. Jest to uproszczony kształt pojazdu samochodowego. Na poniższym rysunku zostały przedstawione jego wymiary.

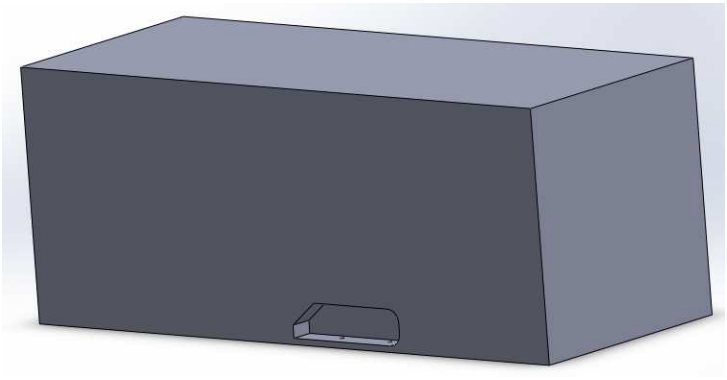


Rys. 23. Ciało Ahmeda[13]

W celu dokonania obliczeń, pierwszym krokiem było stworzenie odpowiedniej domeny. Do jej wykonania, użyty został program SolidWorks w wersji studenckiej.

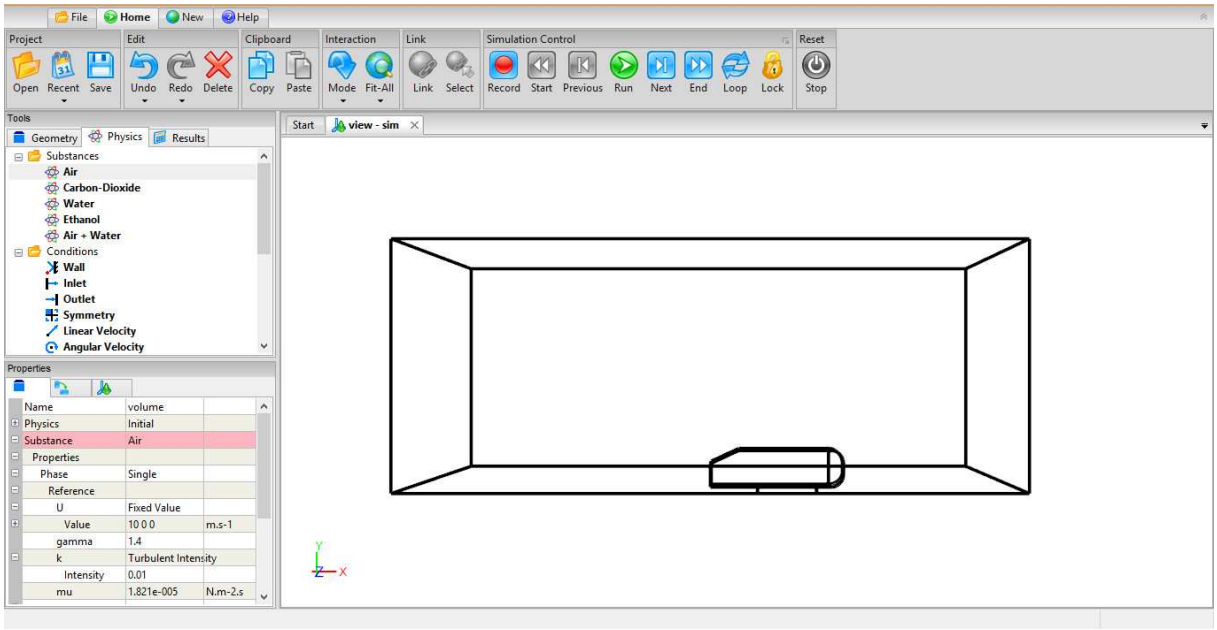


Rys. 24. Model ciała Ahmeda



Rys. 25. Model domeny

Następnie dokonano importu geometrii do środowiska programu Caedium

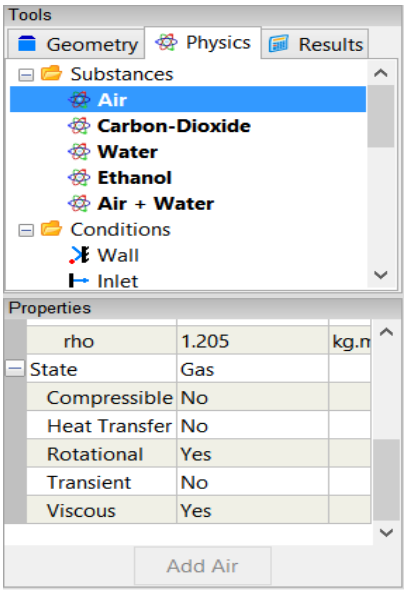


Rys. 26. Importowany model domeny

Program automatycznie odnajduje wszystkie wierzchołki, krawędzie i płaszczyzny podanej geometrii. Korzystając z zakładki „physics” Przygotowano domenę do warunków symulacji, odbyło się to w kilku krokach:

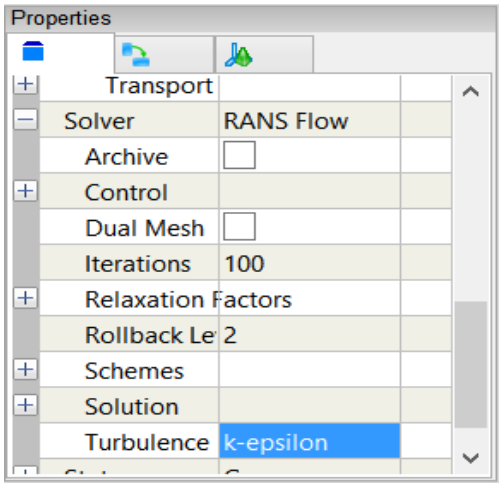
a) nadanie płaszczyznom domeny właściwości ścian, czyli powierzchni przez które nie przepływa płyn, pozwoli to na ograniczenie przepływu tylko do obszaru domeny

b) określenie właściwości płynu w domenie. Program umożliwia definiowanie dowolnego płynu, w tym przypadku skorzystano z wcześniejszych ustawień dla powietrza.



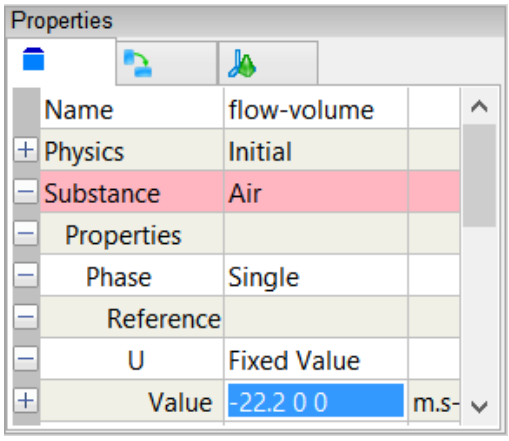
Rys. 27. Określanie własności płynu

c) określenie modelu turbulencji dla metody RANS



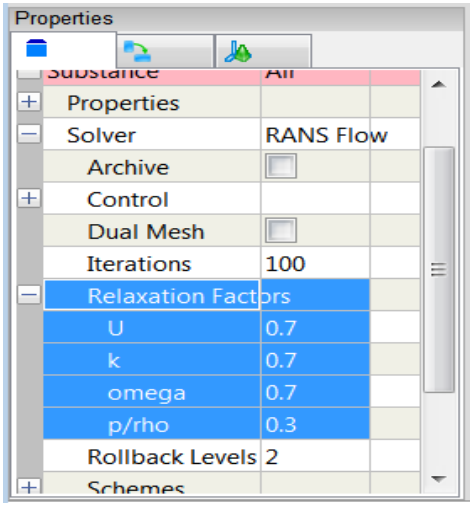
Rys. 28. Określanie modelu turbulencji

d) określenie wartości prędkości płynu w układzie



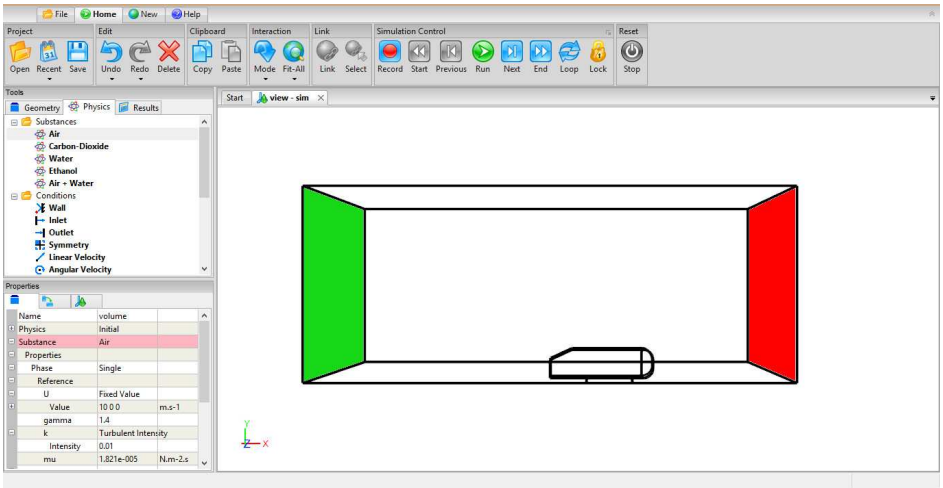
Rys. 29. Określanie wartości prędkości płynu

e) określenie parametrów relaksacji



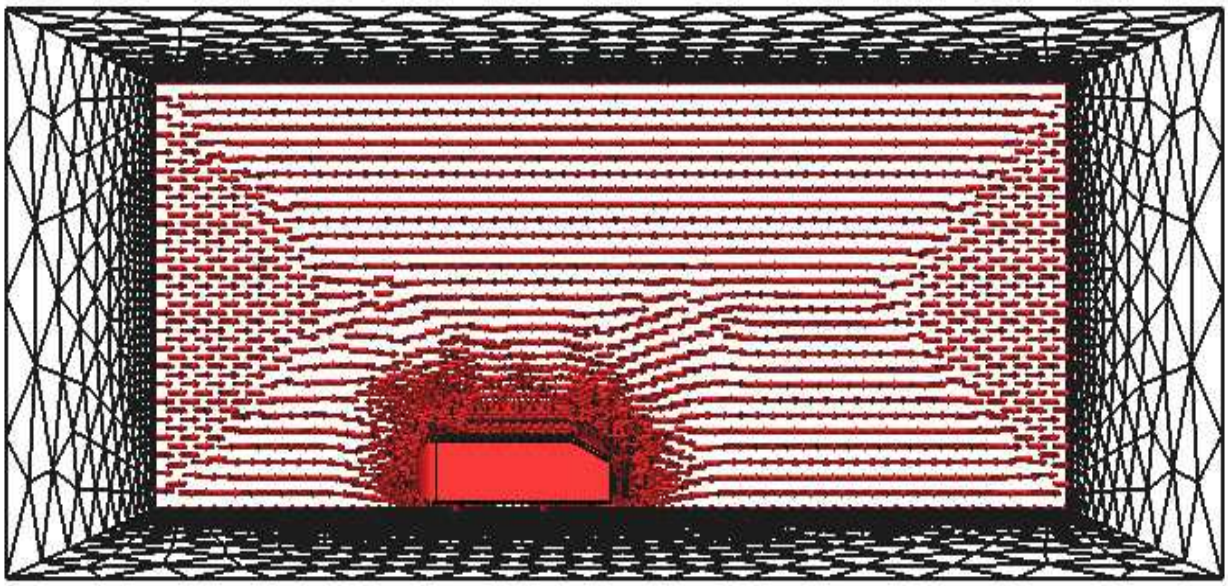
Rys. 30. Określanie parametrów relaksacji

f) określenie wlotu i wylotu płynu (powietrza) z domeny



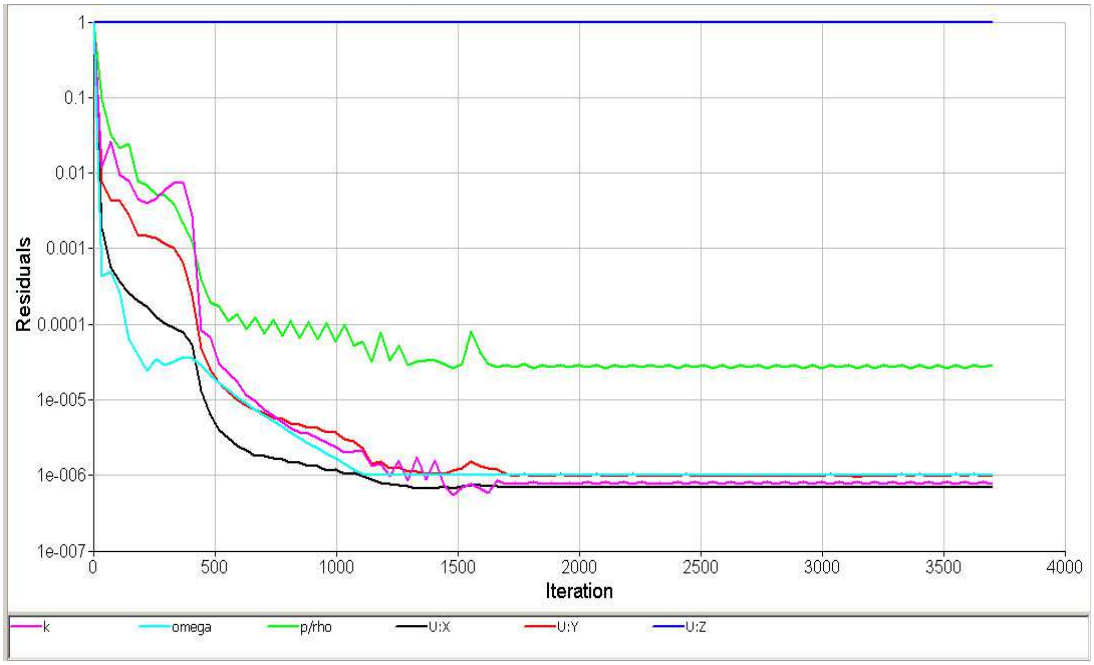
Rys. 31. Oznaczenie wlotu (kolor czerwony) i wylotu (kolor zielony) powietrza

g) generacja siatki i dobór zagęszczenia w określonych miejscach



Rys. 32. Domena z wygenerowaną siatką

h) dołączenie monitora residuów

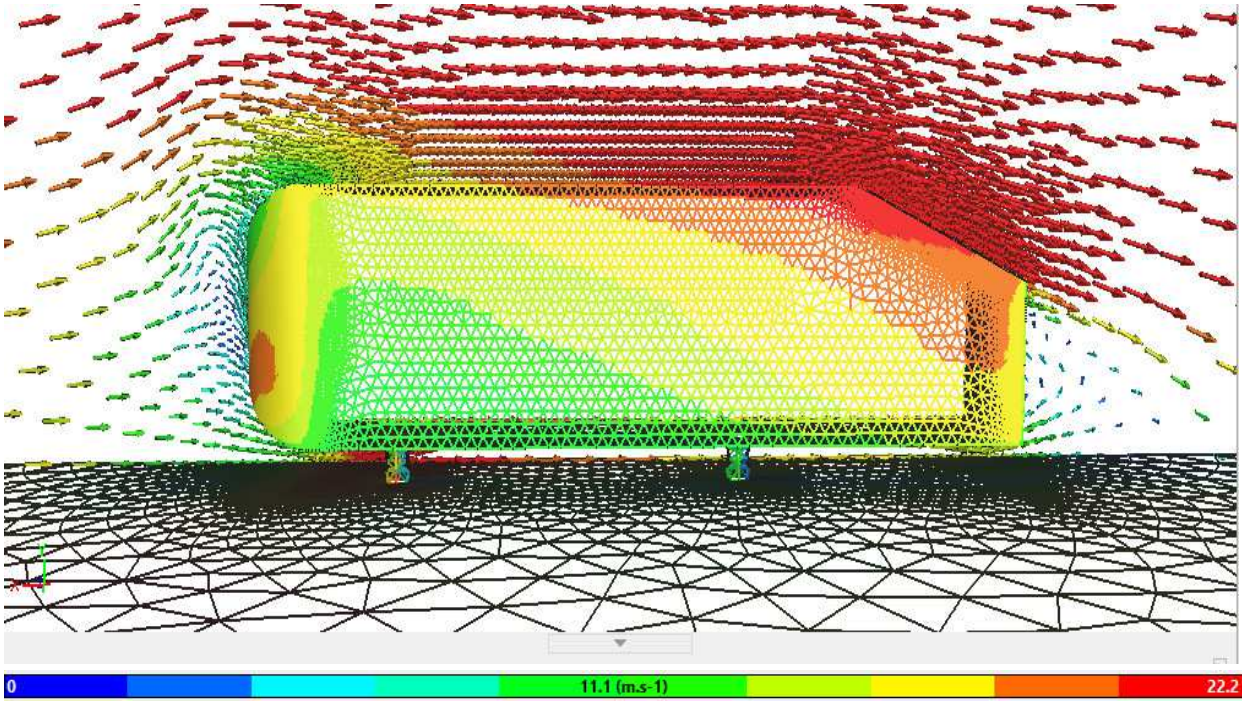


Rys. 33. Monitor residuów

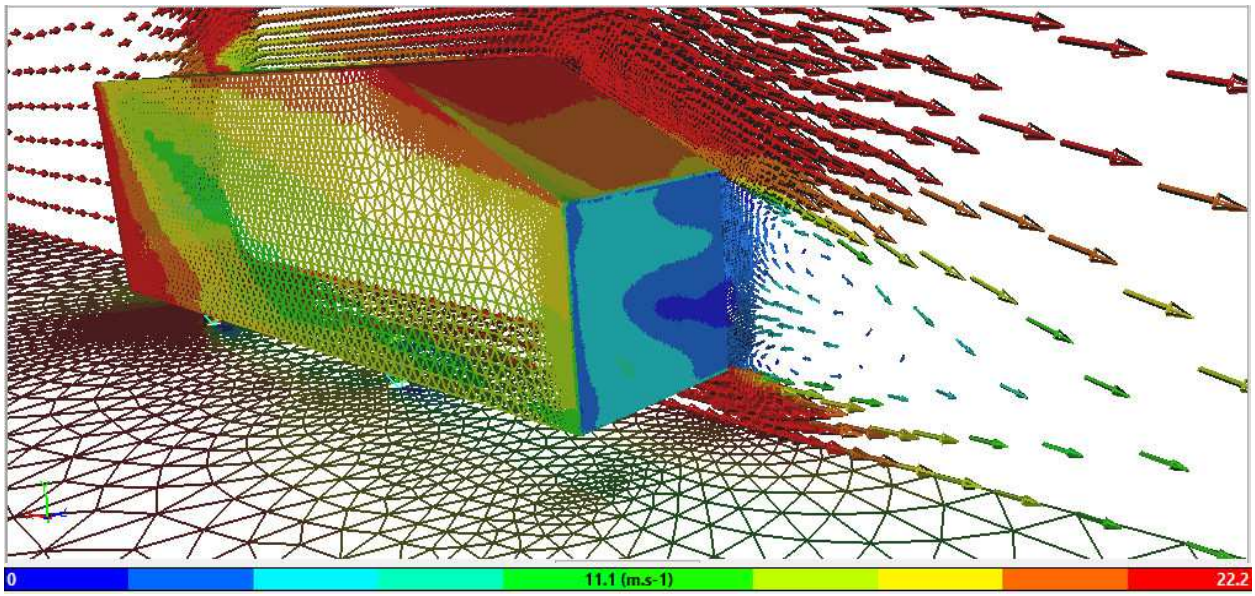
3.8. Wyniki

Symulacje przeprowadzono dla trzech różnych kątów pochylenia tylnej części modelu. Ich wartości wynosiły kolejno: 25, 30 i 40 stopni. Prędkość płynu wynosiła 50km/h, a liczba Reynoldsa 42 miliony. Otrzymane wyniki są rozkładami prędkości.

a) 25 stopni:

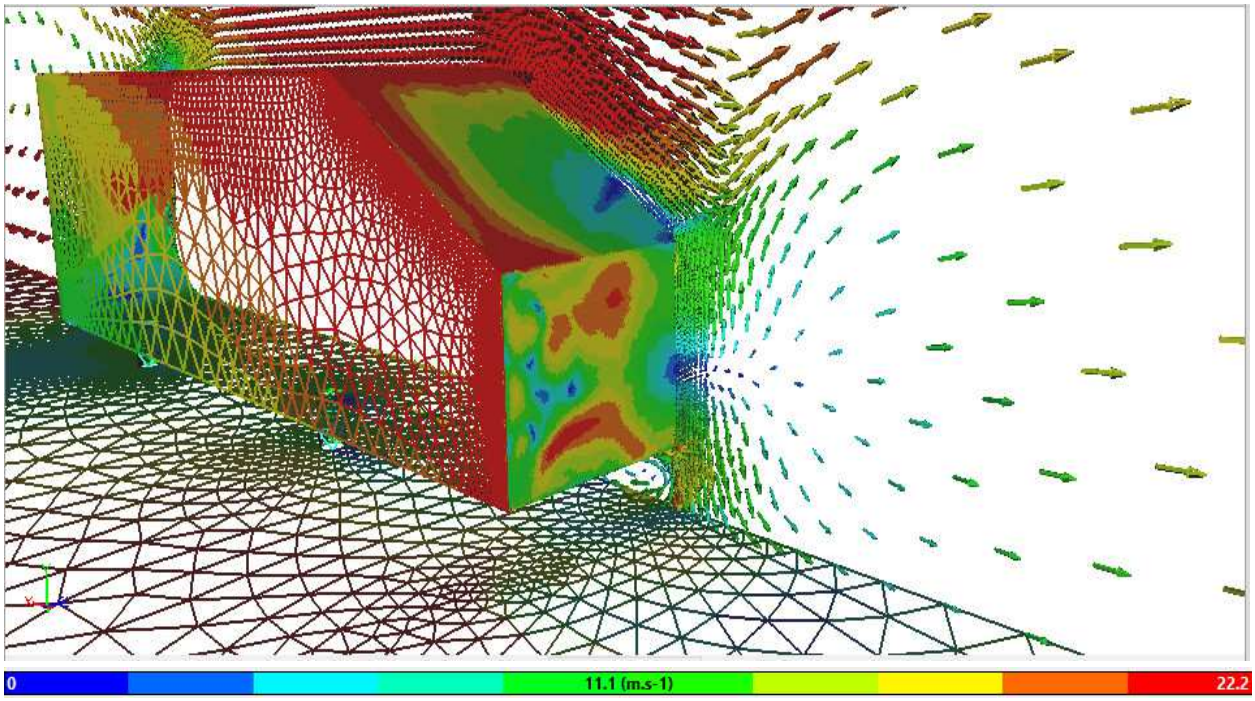


Rys. 34. Widok z boku

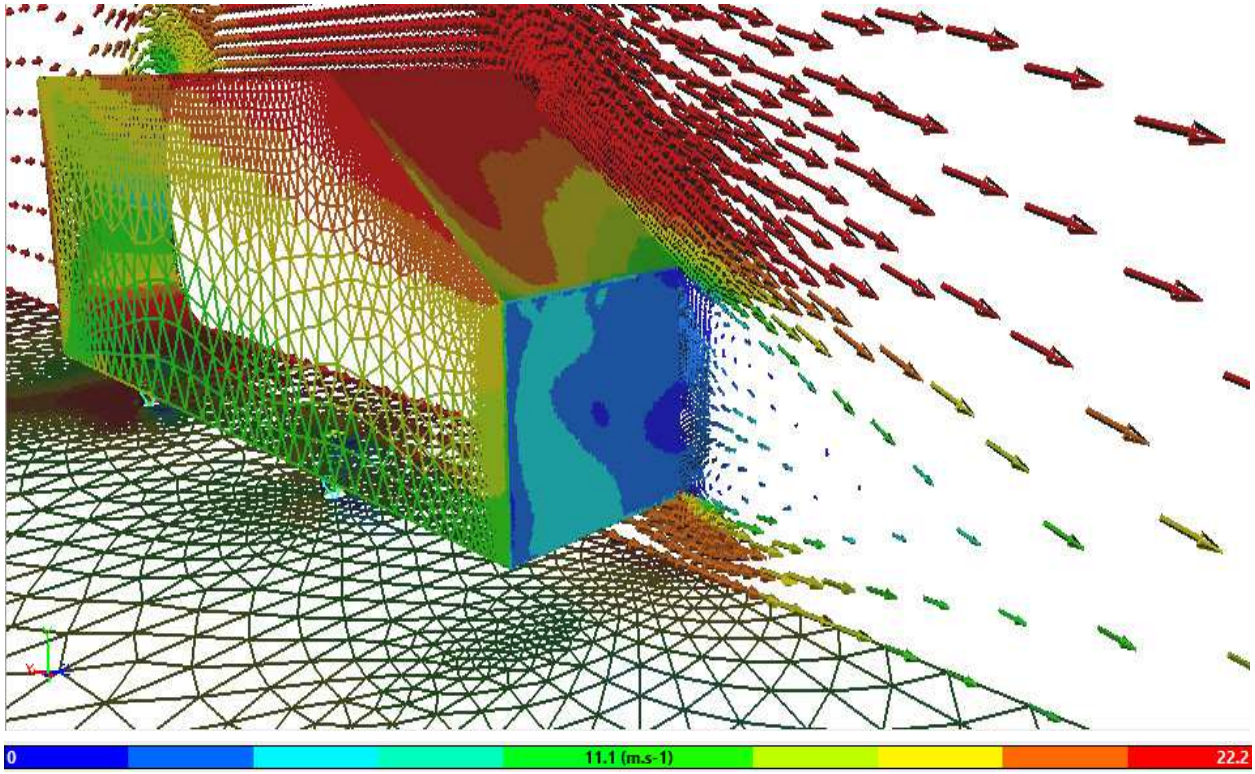


Rys. 35. Widok z tyłu

b) 30 stopni

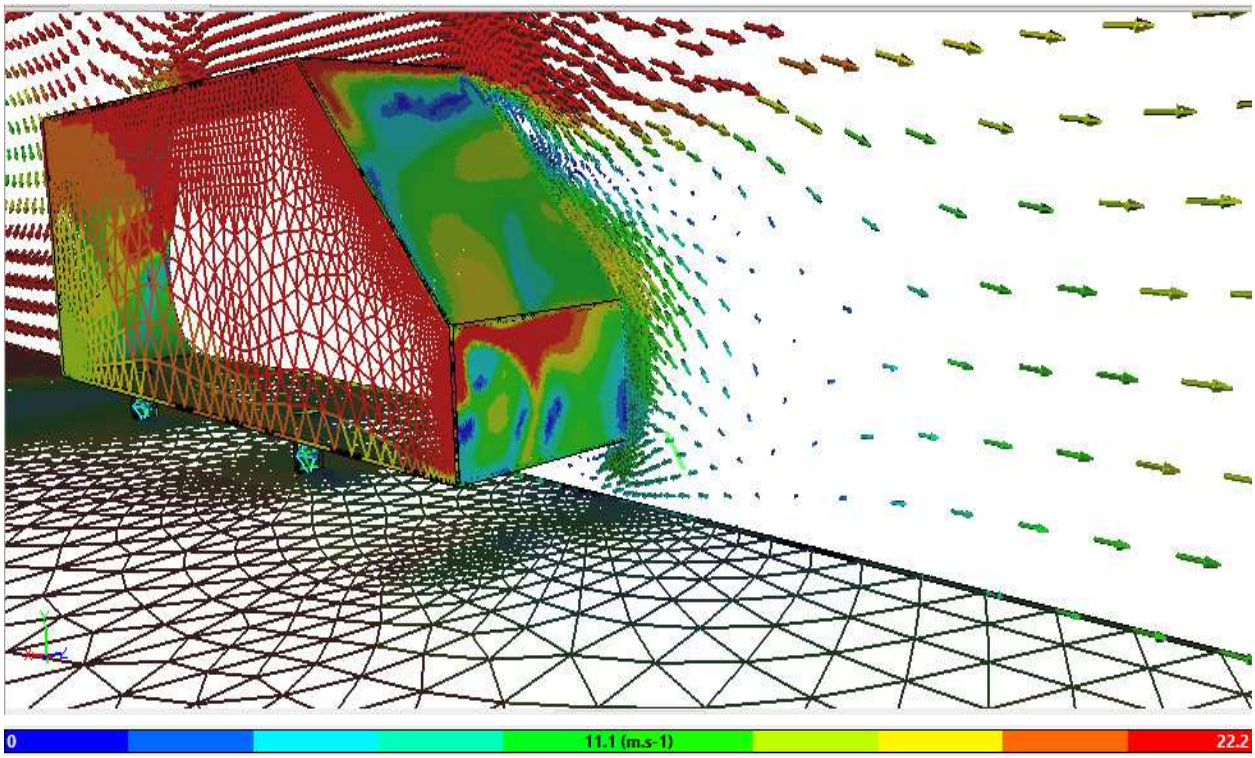


Rys. 36. Widok z tyłu (początek tworzenia się wirów)

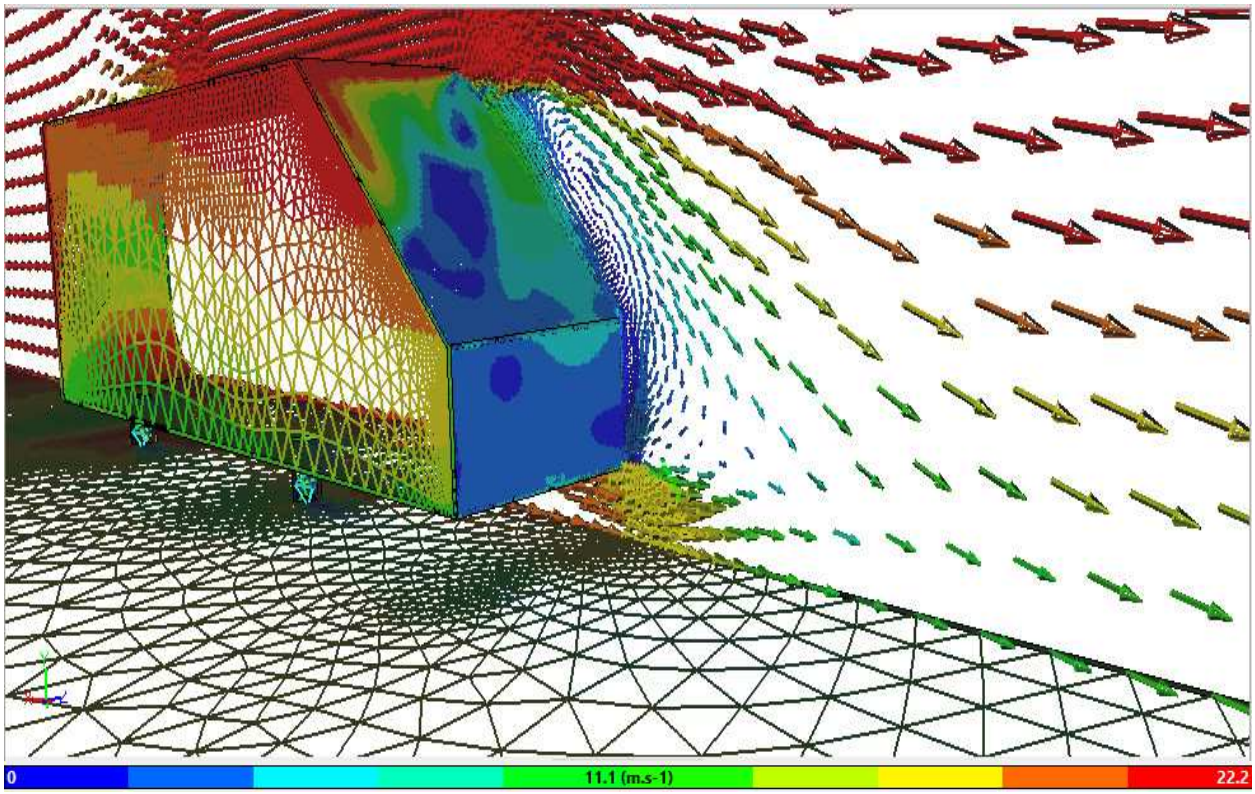


Rys. 37. Monitor residuów (stan ustalony wiru)

c) 40 stopni:



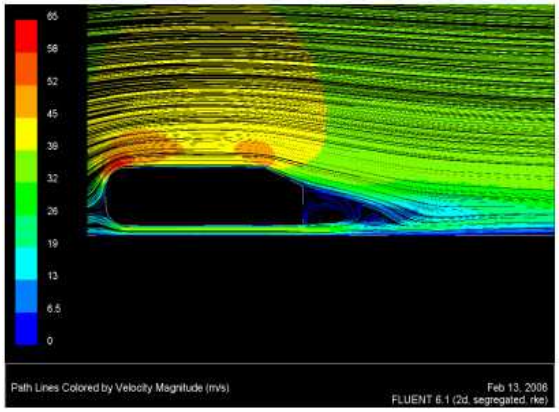
Rys. 38. Widok z tyłu (początek tworzenia się wirów)



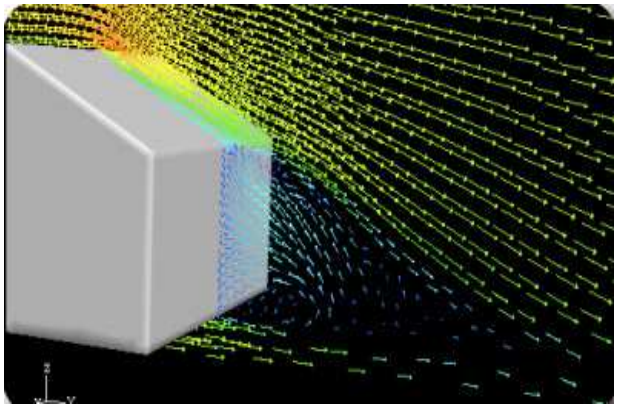
Rys. 39. Widok z tyłu (stan ustalony wiru)

4. PODSUMOWANIE

Założeniem pracy było ukazanie zastosowań technologii Nvidia CUDA, w dziedzinie numerycznej mechaniki płynów. Dzięki zapoznaniu się i zastosowaniu wybranego oprogramowania cel pracy został osiągnięty. Wyniki przeprowadzonych symulacji są poprawne, ponieważ pokrywają się z efektami obliczeń zawartych w wielu publikacjach. Poniżej przedstawiono wybrane z nich:



Rys. 40. Przedstawienie opływu Ciała Ahmeda [13]



Rys. 41. Przedstawienie opływu Ciała Ahmeda [12]

Rozkłady prędkości otrzymane podczas obliczeń są zbliżone do w wyników prac Rajamaniego[13] i Staldera[12]. Różnice w efektach symulacji mogły być spowodowane np. wykorzystaniem innego oprogramowania oraz zagęszczeniem i jakością wygenerowanej siatki .

Do zrealizowania tematu pracy niezbędne było poznanie technologii CUDA, jej struktury, zasady działania i zastosowań. Przydatna okazała się również wiedza pozyskana w toku studiów inżynierskich, z takich dziedzin jak: mechanika płynów i modelowanie 3D. Jednym z kilku problemów było znalezienie odpowiedniego oprogramowania, pozwalającego na przeprowadzenie obliczeń z wykorzystaniem technologii Nvidia CUDA. Otrzymanie poprawnych wyników symulacji, wymagało zapoznania się ze środowiskami wykorzystanego oprogramowania oraz dobraniem odpowiednich parametrów symulacji.

Zagadnienie przedstawione w pracy jest tylko niewielką częścią z zakresu zastosowań i możliwości technologii CUDA. Z powodzeniem odnajduje się w rozwiązywaniu problemów o dowolnym stopniu komplikacji, również w innych dziedzinach nauki.

5. LITERATURA

[1] Crespo AJC, Dominguez JM, Barreiro A, Gómez-Gesteira M and Rogers BD, 2011. *GPUs, a new tool of acceleration in CFD*: Efficiency and reliability on Smoothed Particle Hydrodynamics methods. PLoS ONE. doi:10.1371/journal.pone.0020685.

[2] Czech Z. J, *Obliczenia równoległe,* Politechnika Śląska, 2011 Dostępny: ftp://sun.aei.polsl.pl/pub/zjc/or.pdf

[3] Wiklund K, *Visual Interactive Simulation*: SPH, 2007 Dostępny: https://www8.cs.umu.se/kurser/5DV058/VT10/lectures/Lecture8.pdf

[4] Gómez-Gesteira M, Rogers BD, Crespo AJC, Dalrymple RA, Narayanaswamy M and Dominguez JM (2012a) *SPHysics - development of a free-surface fluid solver*- Part 1: Theory and Formulations. Computers & Geosciences, doi:10.1016/j.cageo.2012.02.029.

[5] Gómez-Gesteira M, Crespo AJC, Rogers BD, Dalrymple RA, Dominguez JM and Barreiro A (2012b) *SPHysics - development of a free-surface fluid solver*- Part 2: Efficiency and test cases. Computers & Geosciences, doi:10.1016/j.cageo.2012.02.028.

[6] Barney B: *Introduction to Parallel Computing*, 2007.

[7] Almasi G.S, Gottlieb A, *Highly Parallel Computing. Benjamin-Cummings publishers*, Redwood city, CA, 1989.

[8] Szpindler M, Cytowski M. *Projektowanie algorytmów równoległych*, Uniwersytet Warszawski, 2009. Dostępny: http://www.icm.edu.pl/c/document_library/get_file?uuid=d114a9a1-bac0-46b9-8dce-a1645b5658e9&groupId=10128

[9] Nvidia Corporation, *CUDA C Programming Guide*: Design Guide, 2013. Dostępny: http://docs.nvidia.com/cuda/cuda-c-programming-guide/

[10] Nvidia Corporation, *CUDA C Best Practices Guide*: Design Guide, 2011. Dostępny: http://www.shodor.org/media/content//petascale/materials/UPModules/dynamicProgrammingCUDAPtII/bestPractices.pdf

[11] Kustner T, Weidendorfer J, and Weinzierl T, *Argument Controlled Profiling*, Proceedings of Euro-Par 2009 – *Parallel Processing Workshops*, Lecture Notes in Computer Science, Vol. 6043, pp. 177-184, 2010

[12] Stalder B. Obe M, *Etude d'un écoulement fluide autour d'un corps de voiture simplifié, Juin Institut National des Sciences Appliqués, 2007*

[13] Rajamani G. K, CFD Analysis of Air Flow Interactions in Vehicle Platoons, RMIT University, 2006

[14] https://developer.nvidia.com/cuda-gpus

[15] http://dual.sphysics.org/

[16] http://www.mechanikryki.pl/renata/pliki_pdf/procesor.pdf

[17] http://www.symscape.com

[18] http://tesla-liczy-cuda.pl/tesla/tesla-sub

[19] http://twimgs.com/ddj/images/article/2008/0812/081229gointelmany1_f2.png

[20] http://emea.kontron.com/enewswire/emea/letter/2007/11/netwire/7_topic.php

[21] http://eandt.theiet.org/news/2013/may/hiv-capsid.cfm?origin=EtOtherNews

6. SPIS ILUSTRACJI