

Symulacja rozprzestrzeniania się dymu



Opiekun projektu:
dr inż. Robert Lubaś

Marcin Sawczuk Daniel Warloch Norbert Pilarek

Informatyka II rok
Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej
Akademia Górnictwo-Hutnicza im. Stanisława Staszica w Krakowie

2019/2020

Spis treści

1 Wstęp	3
2 Teoria	3
2.1 Równania Naviera-Stokesa	3
2.2 Ważne pojęcia	4
Układ dyspersyjny	4
Dym	4
Symulacje CFD dymu	4
3 Realizacja	5
3.1 Założenia projektowe	5
Narzędzie	5
Miejsce	5
Wizualizacja	5
3.2 Ewolucja pomysłu wykonania	5
Python	5
Pyjet	6
Fluidity	6
Wyświetlanie	6
Konsultacje	6
Blender	6
MantaFlow	6
3.3 Ostateczna technologia	7
3.4 Implementacja	7
3.5 Symulacje	11
Pierwsza symulacja	11
Druga symulacja	12
Trzecia symulacja	13
3.6 Wyniki	14
4 Wnioski	18
5 Podsumowanie	18

1 Wstęp

Przepływ powietrza wokół samochodu. Fala po uderzeniu meteorytu. Cyrkulacja krwi w układzie krążenia. Wszystkie te zjawiska łączy obecność płynów: cieczy i gazów. W niniejszej pracy próbujemy poznać naukę poświęconą symulowaniu wyżej wspomnianych i wielu innych zjawisk: obliczeniową mechanikę płynów (CFD – Computational Fluid Dynamics).

Głównym zadaniem projektu było stworzenie trójwymiarowego modelu rozprzestrzeniania się dymu w pomieszczeniu. Model uwzględnia m.in. takie czynniki jak temperatura czy kierunek ruchu powietrza. W tej pracy opracowaliśmy model oraz przedstawiliśmy wyniki symulacji 3D dymu.

Nasza symulacja mimo początkowo innych koncepcji ostatecznie opiera się na Równaniach Naviera-Stokesa. Są to równania różniczkowe cząstkowe drugiego rzędu. Operują na polach prędkości i ciśnień, badając ich ewolucję w czasie. Równania Naviera-Stokesa stanowią bardzo dojrzały, dokładnie zgłębiony dział mechaniki płynów.

2 Teoria

2.1 Równania Naviera-Stokesa

Równania Naviera-Stokesa pozwalają zamodelować szeroki wachlarz zjawisk fizycznych. Aby uprościć zagadnienie potraktujmy powietrze jako nieściśliwy płyn niutonowski. Nieściśliwość płynu oznacza, iż jego gęstość pozostaje stała w całej przestrzeni, co jest słusznym przybliżeniem przy niewielkich prędkościach przepływu.

Wyprowadzenie równań Naviera-Stokesa wykracza poza materiał tego sprawozdania; znacznie bardziej wyczerpujące opracowanie tematu można znaleźć m.in. w [7]. Równania Naviera-Stokesa dla nieściśliwego płynu niutonowskiego mają następującą postać:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla) \mathbf{u} - \frac{1}{d} \nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}$$
$$\nabla \cdot \mathbf{u} = 0$$

gdzie \mathbf{u} – prędkość płynu, d – gęstość płynu, p – ciśnienie, ν – lepkość kinematyczna, \mathbf{f} – wypadkowa sił zewnętrznych działających na płyn.

Pierwsze równanie jest przekształconą drugą zasadą dynamiki Newtona dla płynów i stanowi równanie pędu. Człony tego równania można opisać następująco:

- $\frac{\partial \mathbf{u}}{\partial t}$ pochodna prędkości po czasie
- $-(\mathbf{u} \cdot \nabla) \mathbf{u}$ człon adwekcyjny (konwekcyjny). Jest odpowiedzialny za przemieszczanie się prędkości płynu w przestrzeni z upływem czasu.

- $-\frac{1}{d}\nabla p$ człon ciśnieniowy. Reprezentuje siły występujące wewnątrz płynu na skutek różnicy ciśnień między różnymi obszarami płynu.
- $\nu\nabla^2\mathbf{u}$ Człon lepkościowy. Reprezentuje występujące wewnątrz płynu siły tarcia, zachodzące między obszarami o różnych prędkościach. Znaczenie tych sił determinuje lepkość kinematyczną płynu, oznaczona przez ν .
- \mathbf{f} Siły zewnętrzne. Wypadkowa sił działających na płyn, ale nie mających źródła w samym płynie.

Równanie drugie stanowi warunek nieściśliwości – wymusza, by ilość płynu wpływającego do danego obszaru przestrzeni była równa ilości płynu wypływającego.

2.2 Ważne pojęcia

Układ dyspersyjny

Układ, zwykle koloidalny, złożony z co najmniej dwóch faz, z których przynajmniej jedną stanowi silnie rozdrobniony materiał, rozproszony w drugiej fazie o charakterze ciągłym, zwanej ośrodkiem dyspersyjnym. Obie fazy mogą być dowolne, muszą się jednak różnić między sobą składem lub stanem skupienia.

Dym

Układem dyspersyjnym (koloidalnym), w którym fazą ciąglią (rozpraszającą) jest powietrze, a fazą rozproszoną są cząsteczki substancji; Układy dyspersyjne możemy podzielić wg stanu skupienia faz, dymy zaliczą się do aerosoli; dymy tworzące się w atmosferze z przedostających się do niej stałych produktów np. spalania paliw (z kotłowni, pojazdów) lub pochodzących z innej działalności przemysłowej (jak cementownie), zawierające m.in. cząstki sadzy, popiołu, niespalonego paliwa (aerozole atmosferyczne), powodują skażenie środowiska; dym wytwarza się też celowo, np. w walce zbrojnej (zasłona dymna).

Symulacje CFD dymu

Polegają na komputerowym odtworzeniu danego budynku lub pomieszczenia, a następnie wywołaniu w nim cyfrowego pożaru. Modele 3D pokazują szereg istotnych parametrów, jak na przykład prędkość rozprzestrzeniania dymu, temperaturę oraz efektywność systemów oddymiania. W ten sposób można odwzorować nawet najbardziej niekorzystne scenariusze pożaru. Jest to niezwykle szybka i skuteczna metoda na zweryfikowanie warunków ewakuacji oraz warunków w jakich swoje działania prowadzić będzie jednostka ratowniczo-gaśnicza.

3 Realizacja

3.1 Założenia projektowe

Narzędzie

Staraliśmy się nie używać najczęściej wybieranych programów do przedstawiania symulacji dymu oraz pobierania z tego danych i łatwego ich przedstawienia. Po przejrzeniu dostępnych opracowań zauważylismy, że najczęściej stosowanym narzędziem jest Pyrosim. Mimo widocznych zalet postanowiliśmy przebadać inne dostępne możliwości. W dalszej części przedstawimy drogę jaką przeszliśmy, aż do momentu gdy natrafiłyśmy na narzędzie które jednocześnie sprostało naszym wymaganiom oraz nie posiadało zbyt wysokiego progu wejścia.

Miejsce

Chcieliśmy wykonać projekt, który dotyczy również i nas samych. Miejscem które jest jednym z najczęściej odwiedzanych przez całą naszą trójkę jest nasza uczelnia. Zrozumieliśmy już na samym początku, że nie posiadamy i nie będziemy posiadać sprzętu, który w rozsądny czasie dokona nam potrzebnych obliczeń, aby to mógł być ogromny pożar powodujący zadymienie na tak wielkim obszarze jakim jest nasz kampus AGH. Zabraliśmy się za budynek, w którym najprawdopodobniej najczęściej jesteśmy będąc na uczelni, a chodzi o C2. Nasze trudności spowodowały konieczność kolejnych zmian, o których napisaliśmy dalej.

Wizualizacja

Zależało nam na tym, aby przedstawione wyniki przemawiały jak najlepiej do nas, wszyscy jak się okazało jesteśmy wzrokowcami. Podążając za tym postawiliśmy na opieranie się w swoim projekcie na danych przedstawionych w przestrzeni 3D. Skupiliśmy się na trzech parametrach, które według nas w tym aspekcie są najważniejsze:

- widoczność - do jej przedstawienia użyliśmy widoku roznoszenia się cząstek
- temperatura - zastosowaliśmy wizualizację kolorystyczną, gdzie jaśniejszy kolor oznacza wyższą temperaturę
- gęstość - zastosowaliśmy wizualizację kolorystyczną, gdzie jaśniejszy kolor oznacza większą gęstość

3.2 Ewolucja pomysłu wykonania

Python

Pierwszym pomysłem na realizację symulacji był język Python w wersji 3. Jest to język wysokiego poziomu. Jego największą zaletą jest prostota składni, ogromna dostępność różnego typu bibliotek oraz pakietów do wizualizacji oraz przeprowadzania symulacji. Ponadto jest to żywy język, a więc jest ciągle rozwijany.

Pyjet

Po przejrzeniu dostępnych bibliotek wybraliśmy bibliotekę pyjet. Jednak niepowodzenie w zainstalowaniu jej po wielogodzinnych próbach zmusiła nas do porzucenia tego pomysłu.

Fluidity

Kolejnym naszym odkryciem było dependency o nazwie fluidity, które miało nas wesprzeć w zainstalowaniu wcześniej wspomnianej biblioteki oraz bibliotek z rodziny numpy. Niestety zainstalowanie go zajęło kilkanaście godzin, gdyż wymagane było wiele dodatkowych aplikacji i bibliotek wymaganych do komplikacji naszego celu. Jednak nie oddaliśmy się i po kilku dniach osiągnęliśmy sukces.

Wyświetlanie

Po uruchomieniu kilku przykładów podanych w dokumentacji postanowiliśmy stworzyć coś własnoręcznie. Jednak bardzo słaby kunszt wykonania dokumentacji tej biblioteki oraz brak możliwości wyświetlania wyników sumulacji w modelu 3D skłoniło nas do zatrzymania się i zastanowienia się co robić dalej.

Konsultacje

Po zastanowieniu się postanowiliśmy skonsultować z prowadzącym w celu uzyskania profesjonalnej porady. Przedstawiliśmy mu przebytą przez nas drogę. Doradził nam abyśmy wrócili do wyboru oprogramowania które wykorzystamy do wykonania projektu i jeszcze raz dokonali przeglądu dostępnych na rynku rozwiązań.

Blender

Aby rozwiązać ten problem zamiast realizować symulację i pobierać z niej tablicę wyników, a potem obrazować wyniki na płaszczyźnie 3D, postanowiliśmy realizować ją od razu na modelu 3D. W tym celu zastosowaliśmy framework napisany w naszej bibliotece pyjet w Blenderze. Jednak ponownie bardzo słaba dokumentacja zmusiła nas do rezygnacji z tego pomysłu.

MantaFlow

Po ponownym rozejrzeniu się wśród dostępnych rozwiązań i uwzględnieniu naszych poprzednich komplikacji i wyciągniętych wniosków, z pomocą przyszedł nam framework MantaFlow dostępny w programie Blender od wersji 2.82.

3.3 Ostateczna technologia

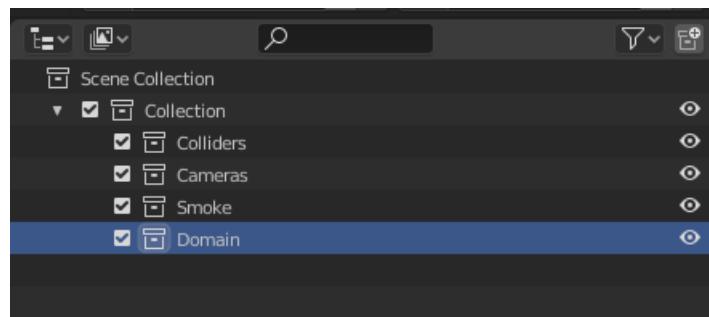
Blender – wolne i otwarte oprogramowanie do modelowania i renderowania obrazów oraz animacji trójwymiarowych

MantaFlow w Blenderze jest open-source frameworkiem symulującym zachowanie płynów w grafice komputerowym i uczeniu maszynowym. Większość jest zaimplementowana w C++, jednak ustawianie parametrów płynu poprzez skrypty jest zaimplementowane w pythonie, a same posługiwania się w niej jest bardzo intuicyjne i przyjemne.

Do interpretacji wyników symulacji wykorzystujemy **MantaFlowGUI**, która także jest bardzo pomocna i łatwa w obsłudze.

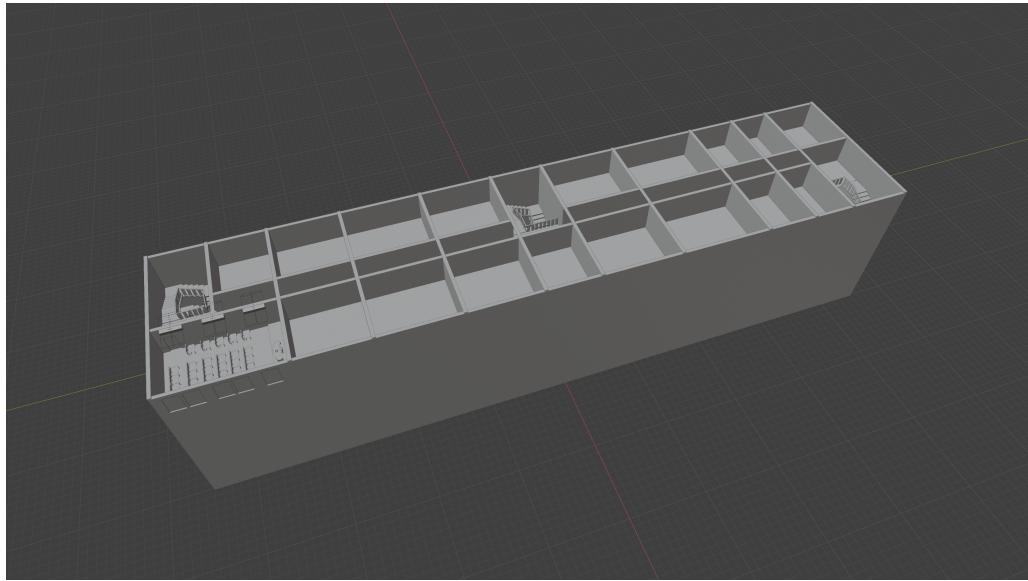
3.4 Implementacja

W programie Blender stworzyliśmy rozwiązanie pozwalające na automatyzację procesu tworzenia symulacji dymu z wykorzystaniem MantaFlow a także skryptów w języku Python. Owe rozwiązanie wykorzystuje 3 kolekcje obiektów w programie Blender do wygenerowania źródła dymu, domeny i effectorów. Natomiast w 4-tej kolekcji umieszczałyśmy kamery z których chcemy uzyskać obraz.

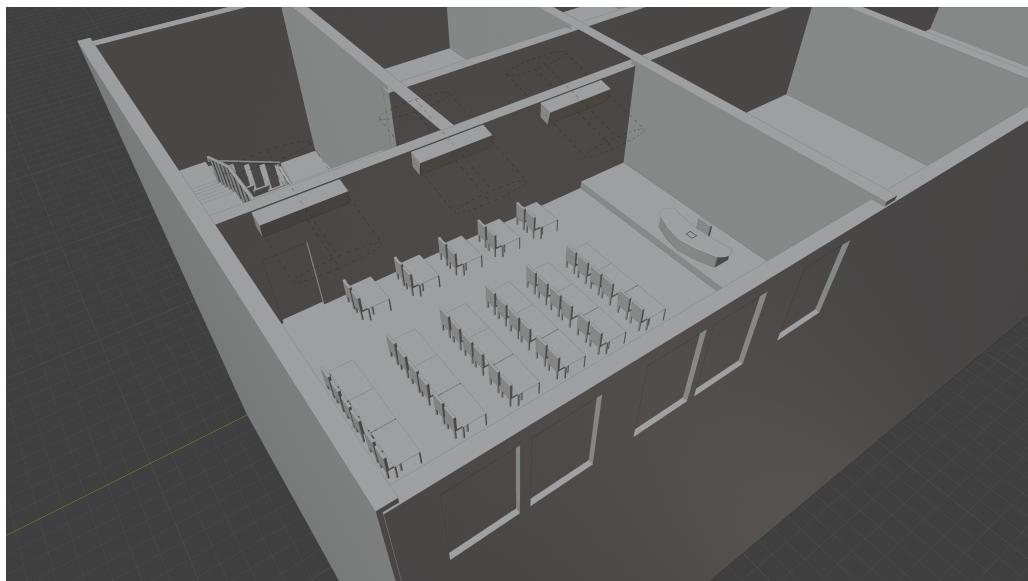


Rysunek 1: Kolekcje obiektów w programie Blender.

Pierwszym przykładem, który chcieliśmy stworzyć był model całego budynku C2, jednak koszt obliczeniowy przeprowadzenia symulacji rozchodzenia się cząsteczek w tej skali był zbyt duży. Ostatecznie naszym punktem startowym i miejscem do którego się ograniczamy jest sala 429.

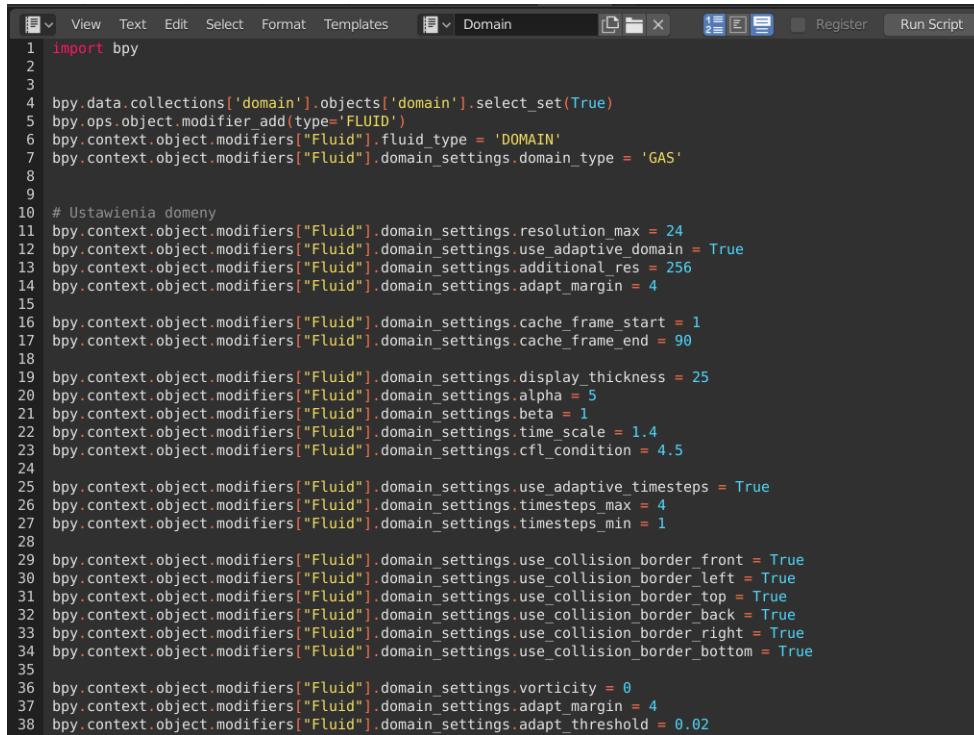


Rysunek 2: Model budynku C2 w Blenderze.



Rysunek 3: Model sali 429 w Blenderze.

Aby nasza symulacja była uniwersalna oraz łatwa w obsłudze do ustawiania parametrów dymu, efektorów i domeny stosujemy skrypty napisane w Pythonie.



```
1 import bpy
2
3
4 bpy.data.collections['domain'].objects['domain'].select_set(True)
5 bpy.ops.object.modifier_add(type='FLUID')
6 bpy.context.object.modifiers["Fluid"].fluid_type = 'DOMAIN'
7 bpy.context.object.modifiers["Fluid"].domain_settings.domain_type = 'GAS'
8
9
10 # Ustawienia domeny
11 bpy.context.object.modifiers["Fluid"].domain_settings.resolution_max = 24
12 bpy.context.object.modifiers["Fluid"].domain_settings.use_adaptive_domain = True
13 bpy.context.object.modifiers["Fluid"].domain_settings.additional_res = 256
14 bpy.context.object.modifiers["Fluid"].domain_settings.adapt_margin = 4
15
16 bpy.context.object.modifiers["Fluid"].domain_settings.cache_frame_start = 1
17 bpy.context.object.modifiers["Fluid"].domain_settings.cache_frame_end = 90
18
19 bpy.context.object.modifiers["Fluid"].domain_settings.display_thickness = 25
20 bpy.context.object.modifiers["Fluid"].domain_settings.alpha = 5
21 bpy.context.object.modifiers["Fluid"].domain_settings.beta = 1
22 bpy.context.object.modifiers["Fluid"].domain_settings.time_scale = 1.4
23 bpy.context.object.modifiers["Fluid"].domain_settings.cfl_condition = 4.5
24
25 bpy.context.object.modifiers["Fluid"].domain_settings.use_adaptive_timesteps = True
26 bpy.context.object.modifiers["Fluid"].domain_settings.timesteps_max = 4
27 bpy.context.object.modifiers["Fluid"].domain_settings.timesteps_min = 1
28
29 bpy.context.object.modifiers["Fluid"].domain_settings.use_collision_border_front = True
30 bpy.context.object.modifiers["Fluid"].domain_settings.use_collision_border_left = True
31 bpy.context.object.modifiers["Fluid"].domain_settings.use_collision_border_top = True
32 bpy.context.object.modifiers["Fluid"].domain_settings.use_collision_border_back = True
33 bpy.context.object.modifiers["Fluid"].domain_settings.use_collision_border_right = True
34 bpy.context.object.modifiers["Fluid"].domain_settings.use_collision_border_bottom = True
35
36 bpy.context.object.modifiers["Fluid"].domain_settings.vorticity = 0
37 bpy.context.object.modifiers["Fluid"].domain_settings.adapt_margin = 4
38 bpy.context.object.modifiers["Fluid"].domain_settings.adapt_threshold = 0.02
```

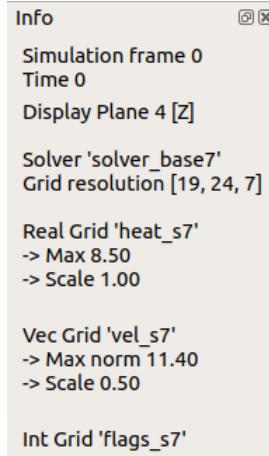
Rysunek 4: Przykładowy fragment skryptu napisanego w Pythonie odpowiedzialny za ustawienie parametrów dymu.

Dodatkowo stworzyliśmy wizualizację kolorystyczną m.in. temperatury i gęstości dymu.



Rysunek 5: Utworzona wizualizacja kolorystyczna.

Aby mieć możliwość pomiaru temperatury, gęstości i innych parametrów dymu w danym punkcie przestrzeni napisaliśmy skrypt pozwalający zwizualizować otrzymane wyniki stosując aplikację MantaFlowGUI, która po zaznaczeniu danego fragmentu dymu dostarcza wszystkich interesujących nas informacji.



Rysunek 6: Przykładowy fragment wyświetlanych danych w MantaFlowGui.

```

setObstacleFlags(flags=flags_s3, phiObs=phiObs_s3, phiOut=phiOut_s3, phiIn=phiIn_s3)
flags_s3.fillGrid()

if timePerFrame_s3 == 0: # Only apply inflow once per frame
    mantaMsg('Smoke inflow at frame: ' + str(framenr))
    applyEmission(flags=flags_s3, target=density_s3, source=densityIn_s3, emissionTexture=emissionIn_s3, type=FlagInflow|FlagOutflow)
    if using_heat_s3:
        applyEmission(flags=flags_s3, target=heat_s3, source=heatIn_s3, emissionTexture=emissionIn_s3, type=FlagInflow|FlagOutflow)

    if using_colors_s3:
        applyEmission(flags=flags_s3, target=color_r_s3, source=color_r_in_s3, emissionTexture=emissionIn_s3, type=FlagInflow|FlagOutflow)
        applyEmission(flags=flags_s3, target=color_g_s3, source=color_g_in_s3, emissionTexture=emissionIn_s3, type=FlagInflow|FlagOutflow)
        applyEmission(flags=flags_s3, target=color_b_s3, source=color_b_in_s3, emissionTexture=emissionIn_s3, type=FlagInflow|FlagOutflow)

    if using_fire_s3:
        applyEmission(flags=flags_s3, target=fuel_s3, source=fuelIn_s3, emissionTexture=emissionIn_s3, type=FlagInflow|FlagOutflow)
        applyEmission(flags=flags_s3, target=react_s3, source=reactIn_s3, emissionTexture=emissionIn_s3, type=FlagInflow|FlagOutflow)

    mantaMsg('Smoke step / s3.frame: ' + str(s3.frame))
    if using_fire_s3:
        process_burn_3()
        smoke_step_3()
    if using_fire_s3:
        update_flame_3()

s3.step()

```

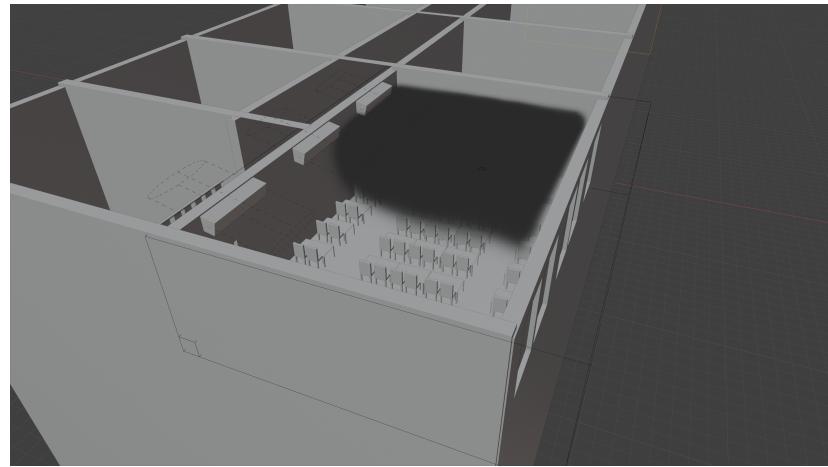
Rysunek 7: Przykładowy fragment skryptu odpowiedzialnego za wyświetlanie parametrów w MantaFlowGui.

3.5 Symulacje

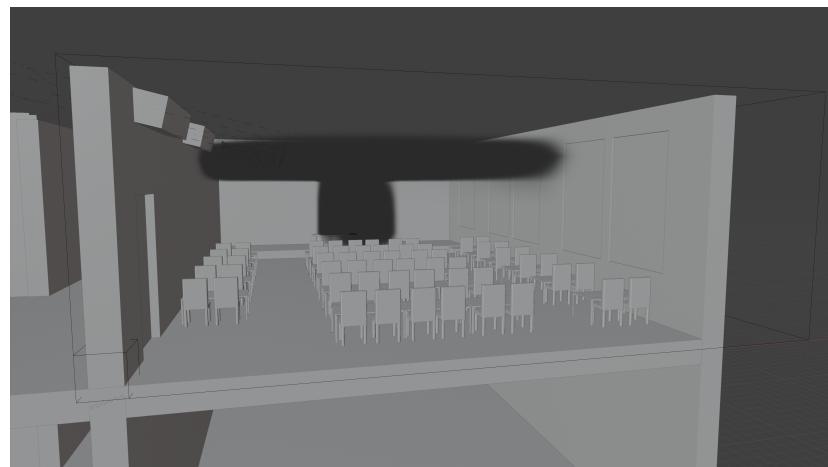
Jak już wcześniej wspomnieliśmy, nasza przykładowa symulacja ma miejsce w sali 429.

Pierwsza symulacja

Na biurku wybucha pożar, z którego dym rozprzestrzenia się po całym pomieszczeniu.

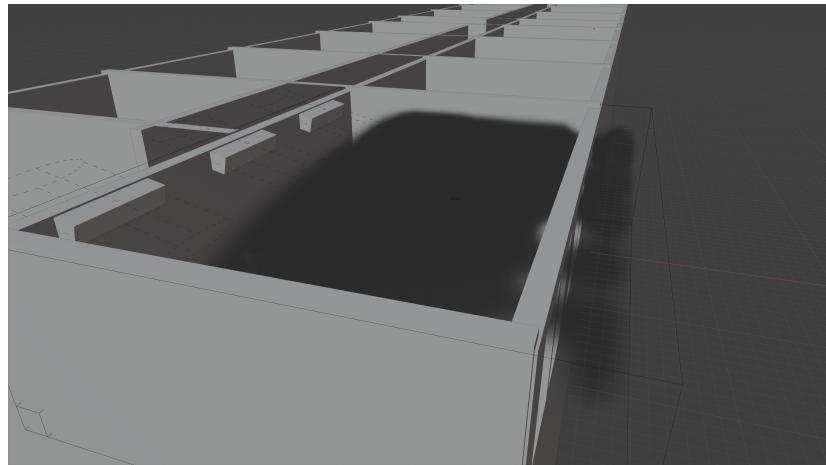


Rysunek 8: Pierwsza symulacja.



Rysunek 9: Pierwsza symulacja.

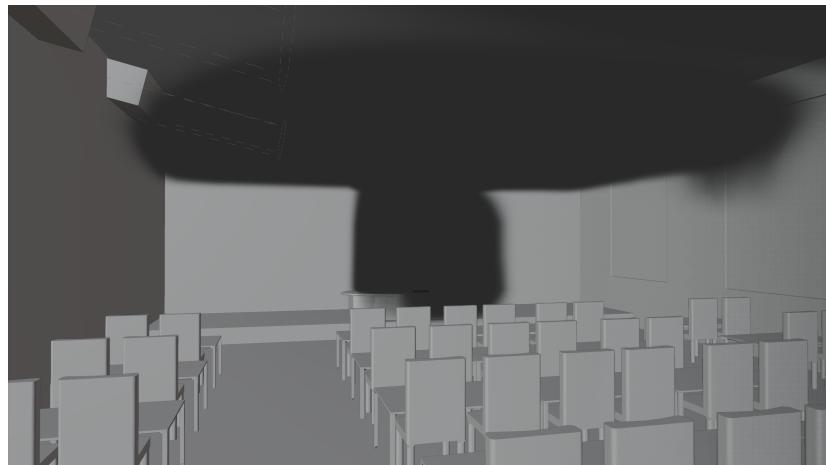
W 230 klatce następuje "ugaszenie pożaru", otworzenie wszystkich okien oraz włączenie systemu oddymiania.



Rysunek 10: Pierwsza symulacja.

Druga symulacja

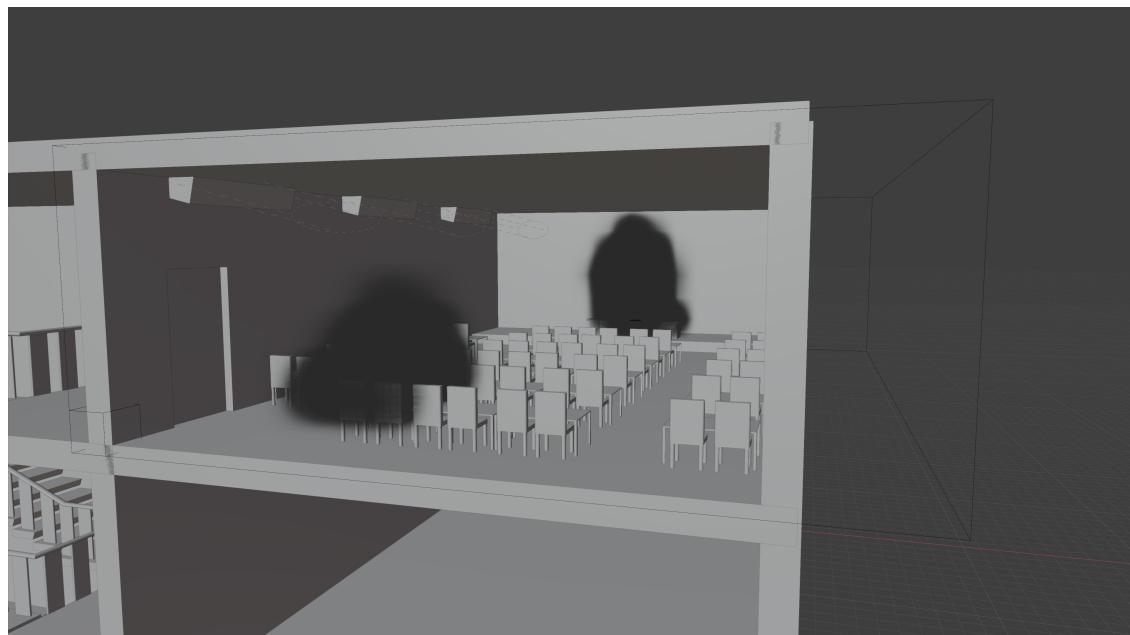
W tym wariancie temperatura emitera jest 4 razy większa niż w poprzednim punkcie, a reszta założeń jest taka sama.



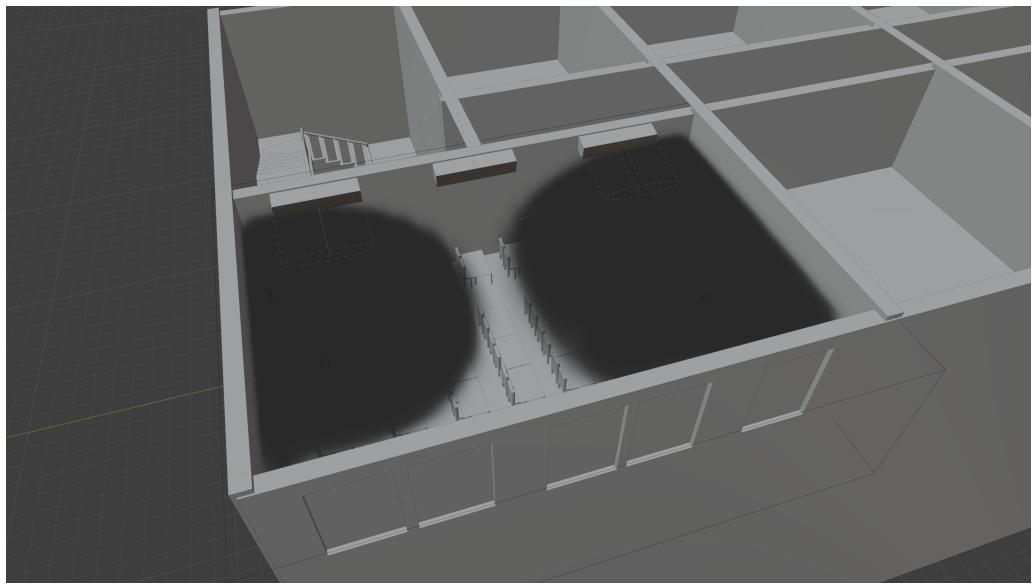
Rysunek 11: Druga symulacja.

Trzecia symulacja

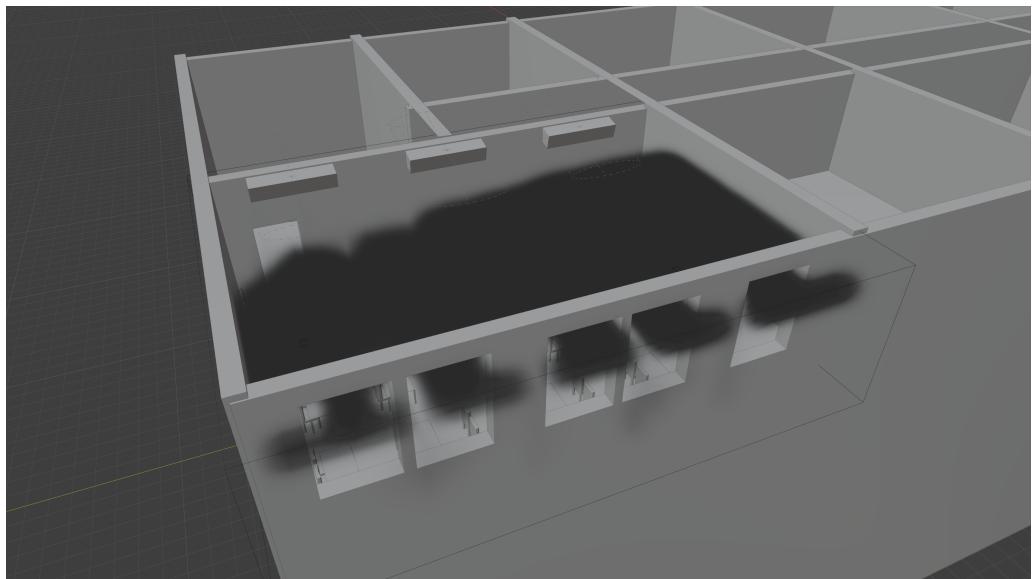
W tym modelu występują 2 źródła dymu, jedno na biurku, jak w wcześniejszych wariantach, a drugie na przeciwnym końcu sali. Źródło znajdujące się na biurku ma 16 razy większą temperaturę emitera, a reszta założeń jest identyczna z wcześniejszymi punktami.



Rysunek 12: Trzecia symulacja.

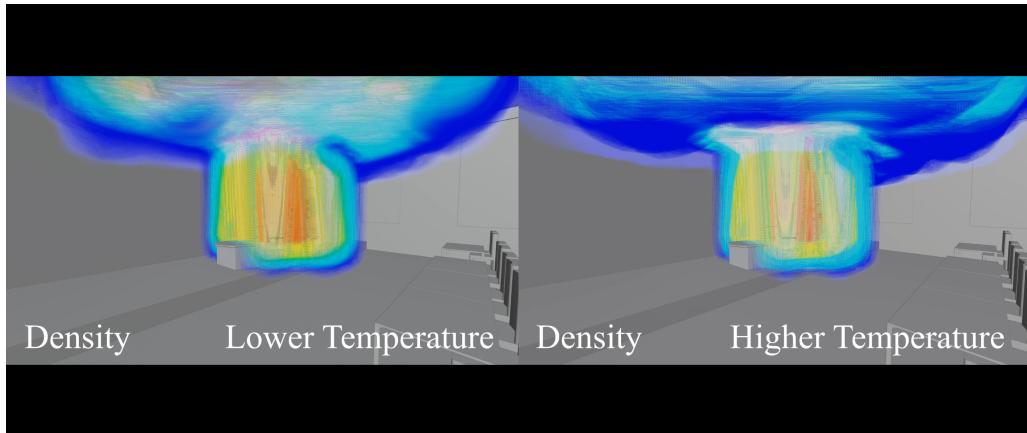


Rysunek 13: Trzecia symulacja.

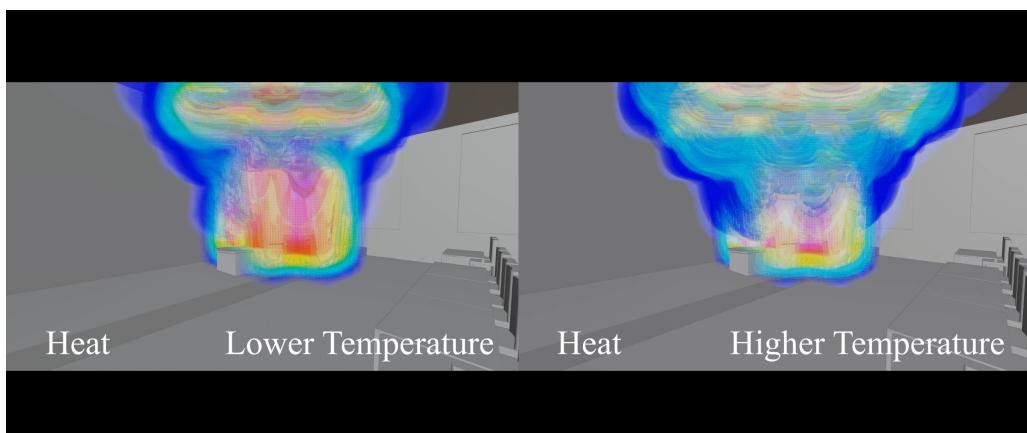


Rysunek 14: Trzecia symulacja.

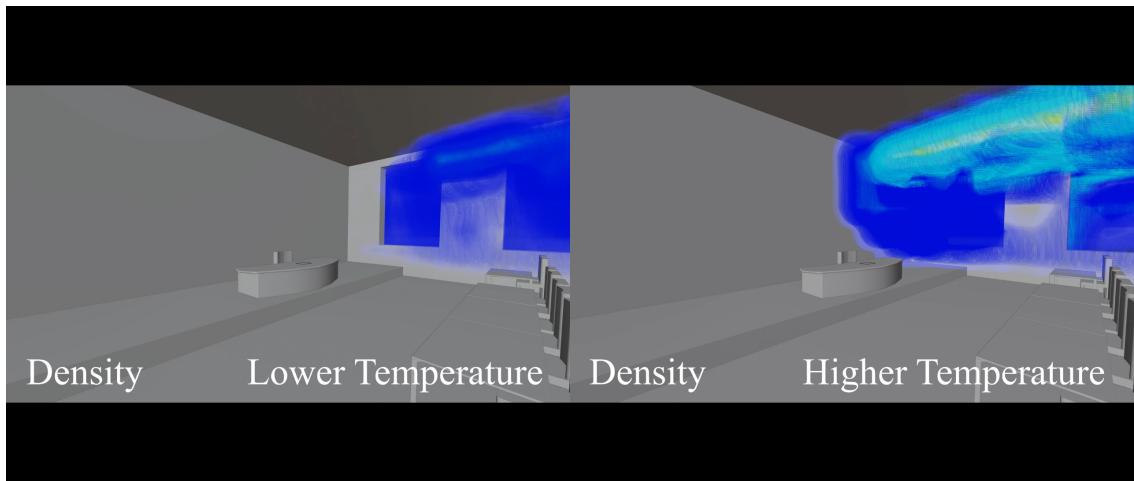
3.6 Wyniki



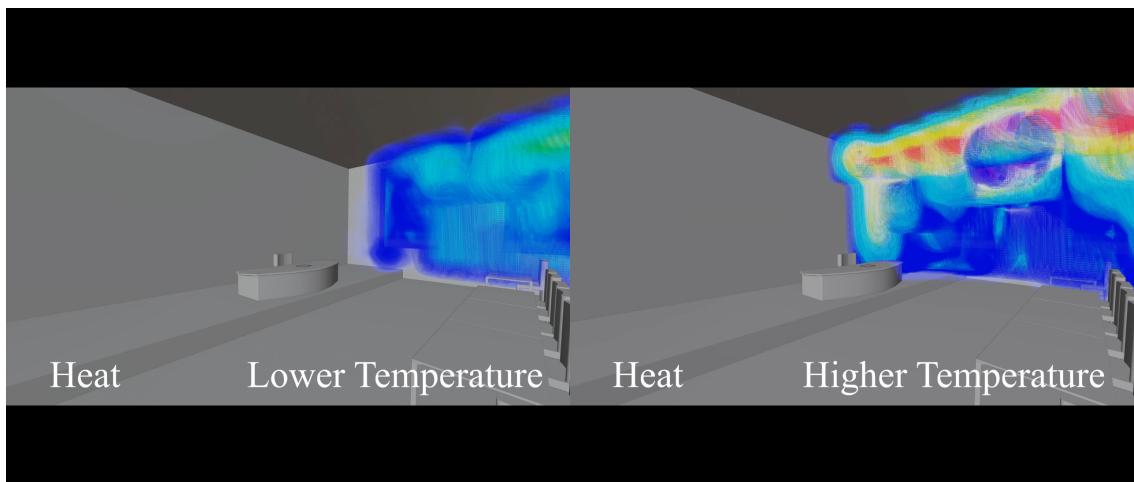
Rysunek 15: Porównanie gęstości dymu przy różnych temperaturach źródła zadymienia.



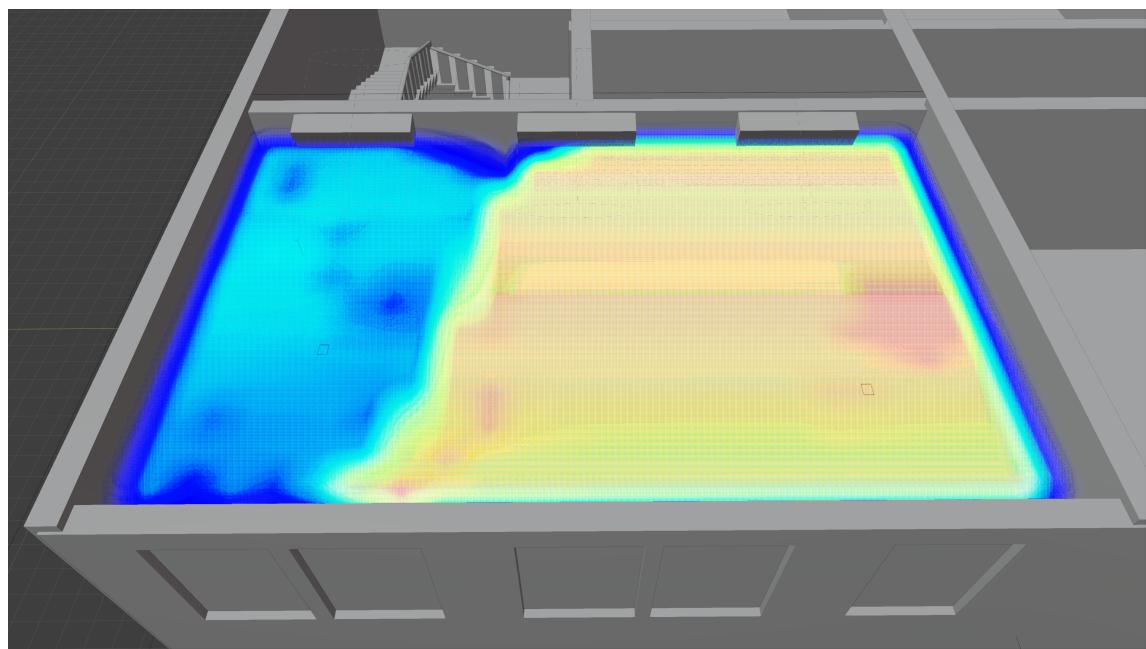
Rysunek 16: Porównanie ciepła dymu przy różnych temperaturach źródła zadymienia.



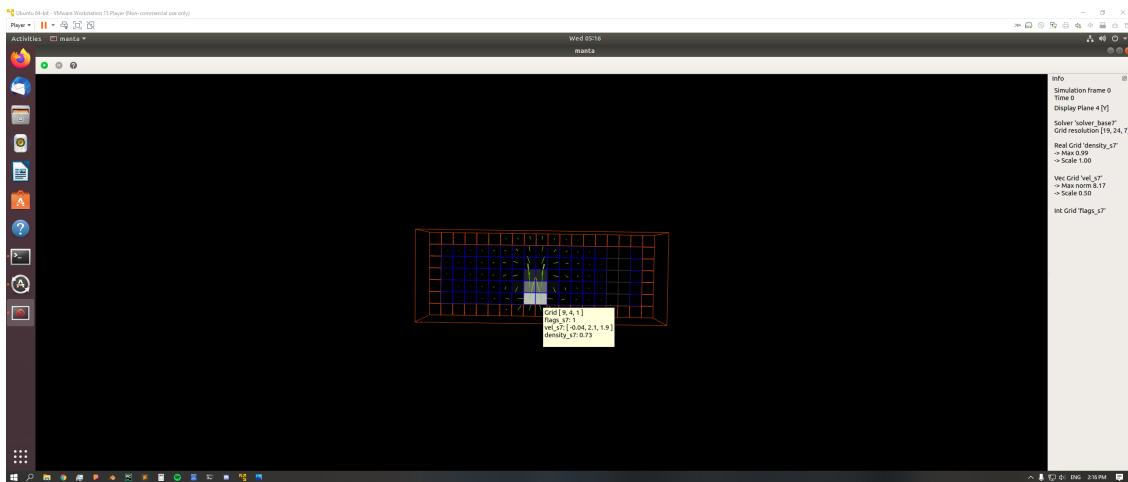
Rysunek 17: Porównanie gęstości dymu przy różnych temperaturach źródła zadymienia.



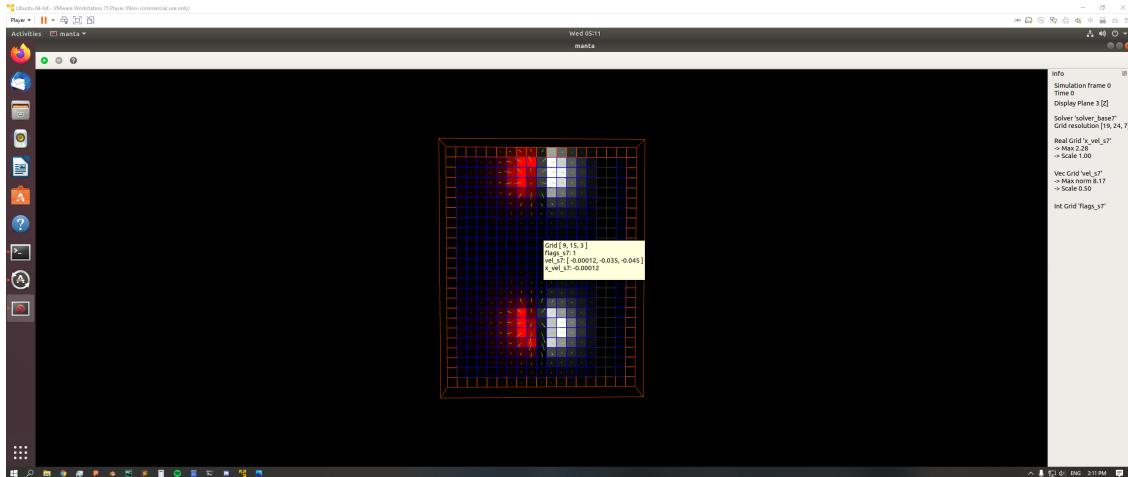
Rysunek 18: Porównanie ciepła dymu przy różnych temperaturach źródła zadymienia.



Rysunek 19: Starcie się dymów o różnych temperaturach źródła zadymienia.



Rysunek 20: Gęstość dymu.



Rysunek 21: Prędkość cząstek.

4 Wnioski

Z naszych symulacji można wyciągnąć następujące wnioski:

4.1 Co doświadczalnie na podstawie uzyskanych wyliczeń i obrazów uzyskaliśmy i potwierdziliśmy

- Im większa temperatura dymu, rozprzestrzenia się on szybciej.
- Im cieplejszy dym, tym unosi się on wyżej, przez co czas wietrzenia jest dłuższy - w naszym modelu przy temperaturach w stosunku 1:16, czas wywietrzania wynosił 1.5 raza dłużej
- Czas wywietrzania jest zależny od sumarycznej mocy oraz rozmieszczenia wentylatorów
- Zatrzymanie źródła dymu nie niweluje zagrożenia w całości, gdyż w sali on się nadal utrzymuje
- Im większa temperatura dymu, tym mniejsza jego gęstość

4.2 Z jakimi głównymi problemami się zmagaliśmy

- Bardzo ciężko stworzyć wirtualny model idealnie odwzorowujący rzeczywiste sytuacje
- Symulacje są bardzo kosztowne obliczeniowo

5 Podsumowanie

Nasze przykładowe symulacje jedynie częściowo obejmują całą złożoność problemu rozprzestrzenia się dymu, jednak stworzony przez nas uniwersalny system modelowania rozchodzenia się dymu jest pełen możliwości - lecz nie jest to jeszcze projekt kompletny i zamknięty.

Ciekawym kierunkiem rozwoju naszego projektu byłoby dodanie różnorodnych skryptów pozwalających na obliczanie oraz prezentowanie statystyk m.in zakresu widoczności w funkcji odległości przy różnym poziomie zadymienia pomieszczenia, wartości średniej temperatury powietrza w zależności od wysokości jej pomiarów.

Przede wszystkim należy jednak rozważyć zadania, jakie miałby realizować program. Wymagania wobec symulatora CFD będą różnić się w zależności, czy badamy rozwój pożaru w budynku, czy tworzymy animację dymu na potrzeby grafiki komputerowej. W jednym przypadku priorytetem może być wierność fizyce, a w drugim wizualna atrakcyjność.

Literatura

- [1] Wikipedia [online] [dostęp: 08.04.2020]. Dostępny w Internecie, link: "układ dyspersyjny Wikipedia"
- [2] Encyklopedia PWN [online] [dostęp: 08.04.2020]. Dostępny w Internecie, link: "dym definicja PWN"
- [3] Ronald Fedkiw, Jos Stam, Henrik Wann Jensen: *Visual Simulation of Smoke*
- [4] Nick Foster and Dimitris Metaxas: *Modeling the Motion of a Hot, Turbulent Gas*
- [5] Eren Algan: *REAL-TIME SMOKE SIMULATION*
- [6] Marinus Rorbech: *REAL-TIME SIMULATION OF SMOKE USING GRAPHICS HARDWARE*
- [7] Ryszard Gryboś: *Podstawy mechaniki płynów. Część 1*
- [8] Lorena A. Barba, Gilbert F. Forsyth: *12 steps to Navier-Stokes*
- [9] Jos Stam: *Stable Fluids*
- [10] David Potter: *Metody obliczeniowe fizyki*
- [11] David Cline, David Cardon, Parris K. Egbert: *Fluid Flow for the Rest of Us: Tutorial of the Marker and Cell Method in Computer Graphics*
- [12] <https://pypi.org/project/pyjet/>
- [13] <https://numpy.org/>
- [14] <https://fluidityproject.github.io/>
- [15] <https://fluidenginedevelopment.org/pdoc/>
- [16] <http://mantaflow.com/>
- [17] <http://mantaflow.com/gui.html>
- [18] <https://www.blender.org/>
- [19] <https://docs.blender.org/manual/en/latest/>
- [20] <https://www.nvidia.com/pl-pl/>
- [21] KRYSYNA JEŻOWIECKA-KABSCH, HENRYK SZEWCZYK: *MECHANIKA PŁYNÓW*