

Przygotowanie projektu w Visual Studio

Na początku należy upewnić się, że na komputerze zainstalowany jest NVIDIA CUDA Toolkit. Tutaj będziemy używali wersji 12.6.

Instrukcja dla Microsoft Visual Studio 2022.

1. Tworzymy nowy projekt

Rozpocznij



Klonuj repozytorium

Pobierz kod z repozytorium online, takiego jak GitHub lub Azure DevOps



Otwórz projekt lub rozwiązanie

Otwórz lokalny plik projektu lub sln programu Visual Studio



Otwórz folder lokalny

Nawiguj i edytuj kod w dowolnym folderze

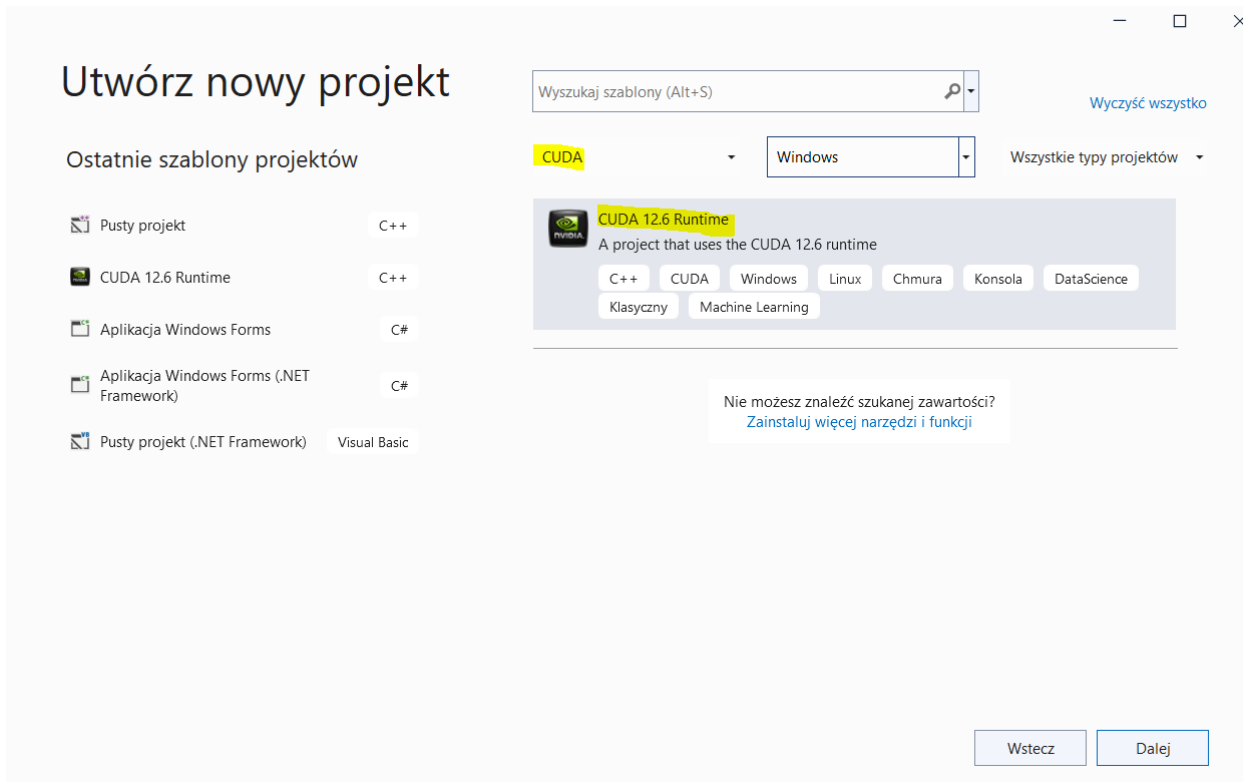


Utwórz nowy projekt

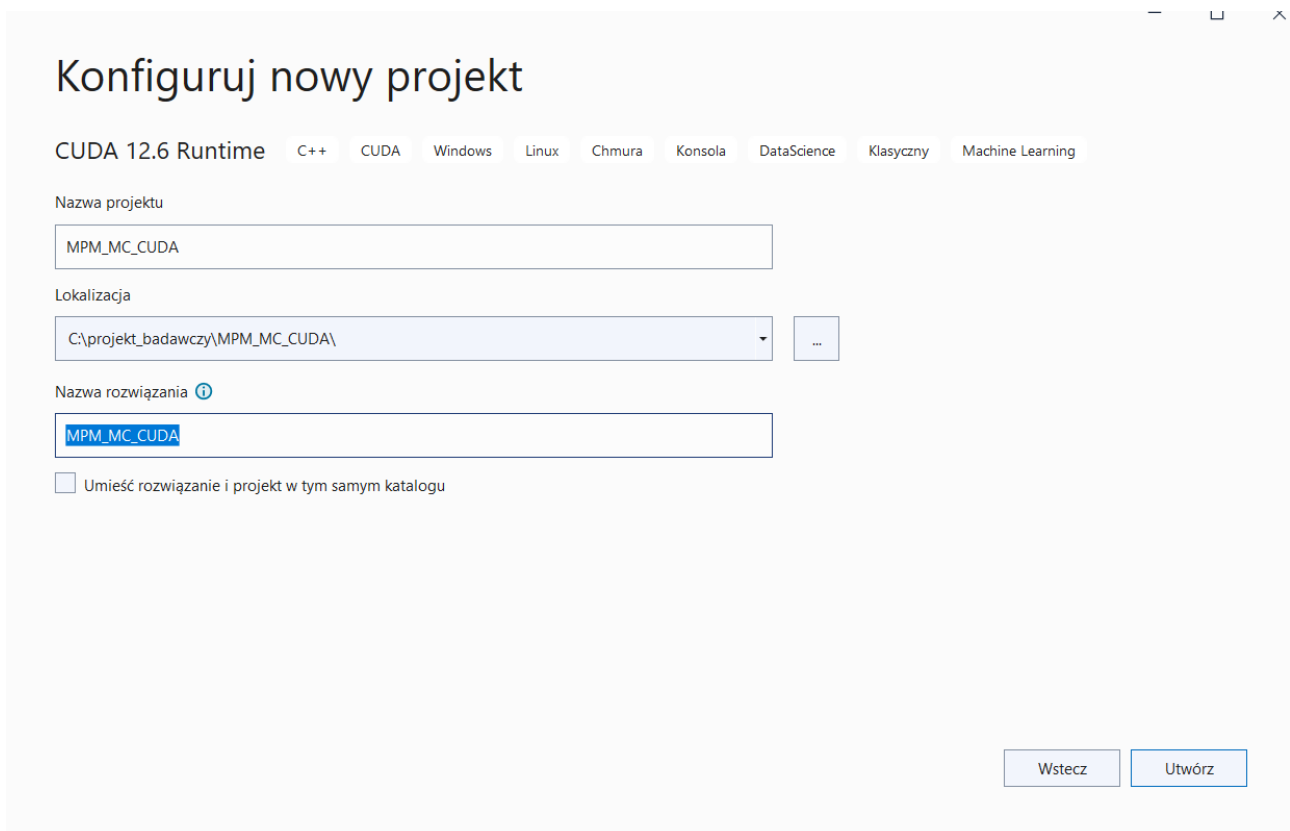
Wybierz szablon projektu z tworzeniem szkieletu kodu, aby rozpocząć pracę

[Kontynuuj bez kodu →](#)

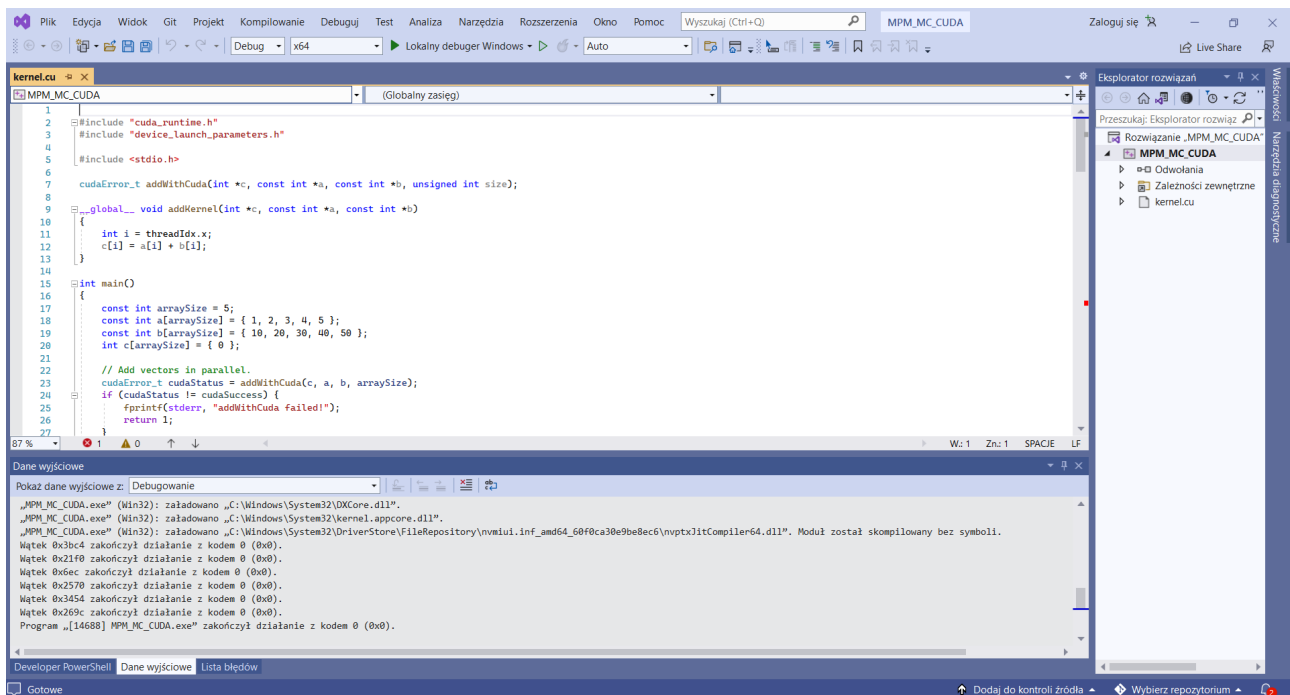
2. Wybieramy szablon dla projektu CUDA



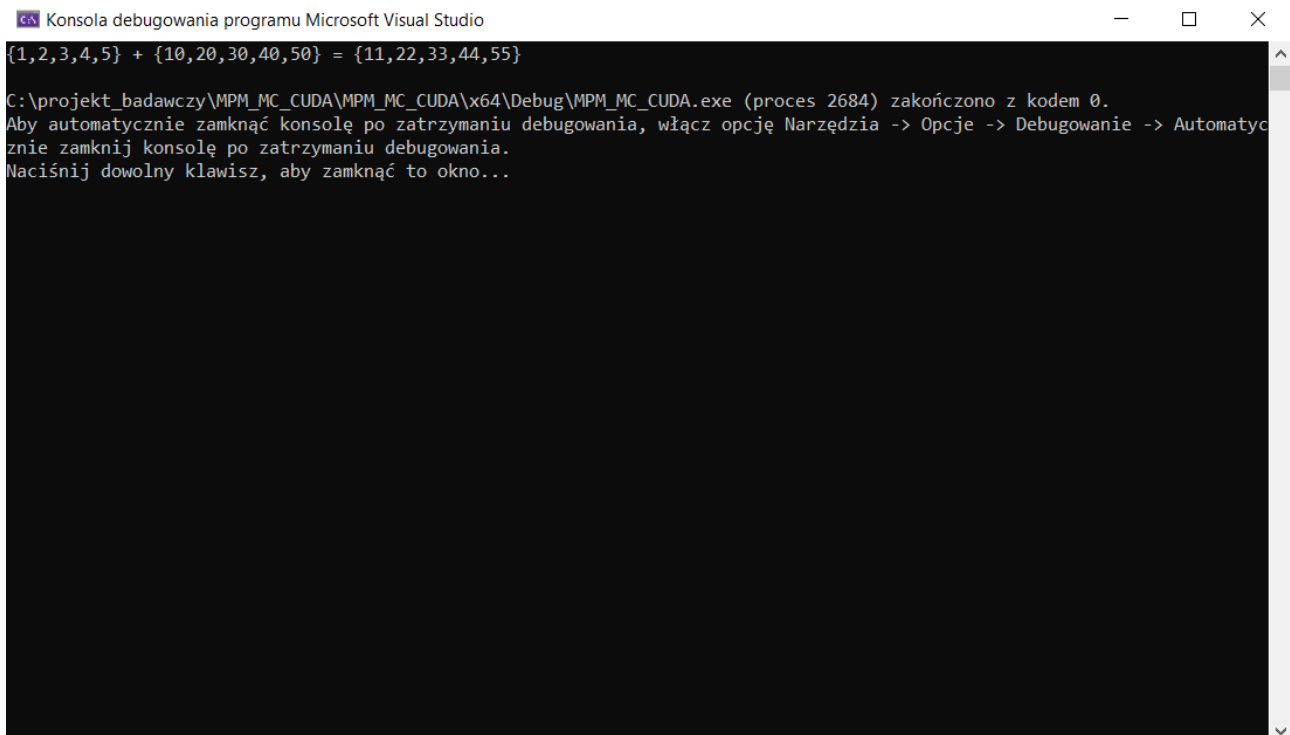
3. Wybieramy ścieżki



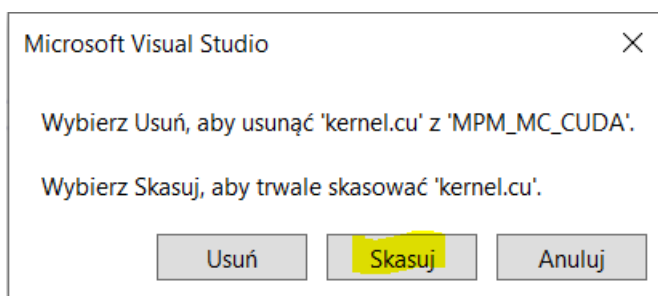
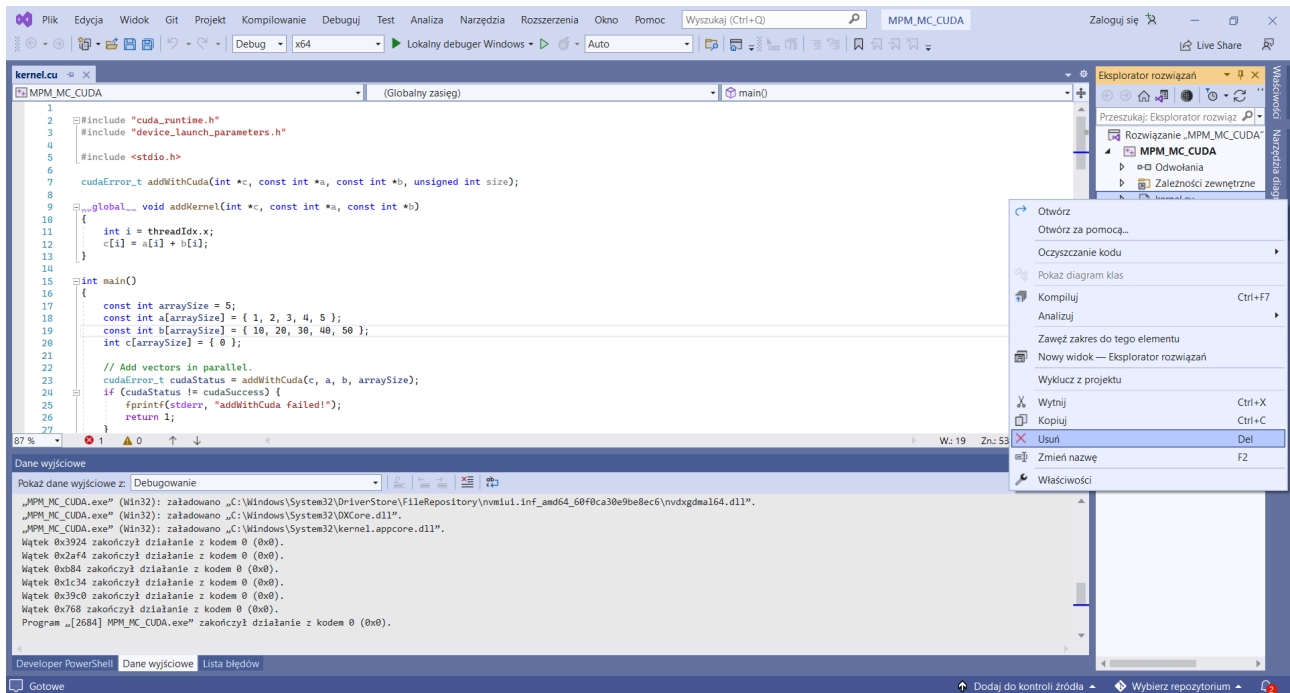
4. Projekt został utworzony, powinniśmy zobaczyć przykładowy plik, pozwalający określić, czy karta graficzna jest wykrywana i przetestować działanie CUDA dla prostego przypadku



5. Jeśli skompilujemy projekt, wynik w terminalu powinien wyglądać następująco

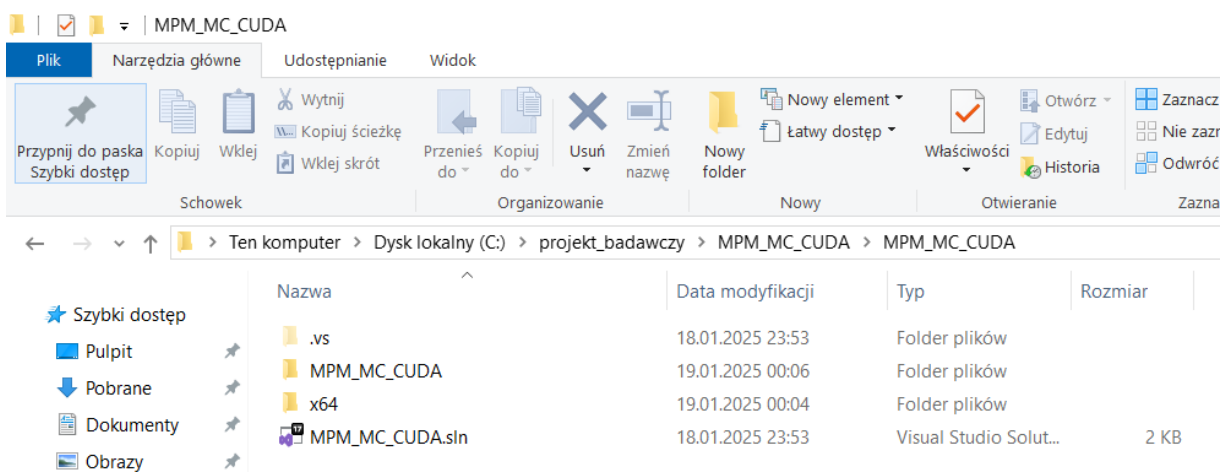


6. Teraz usuniemy plik kernel.cu



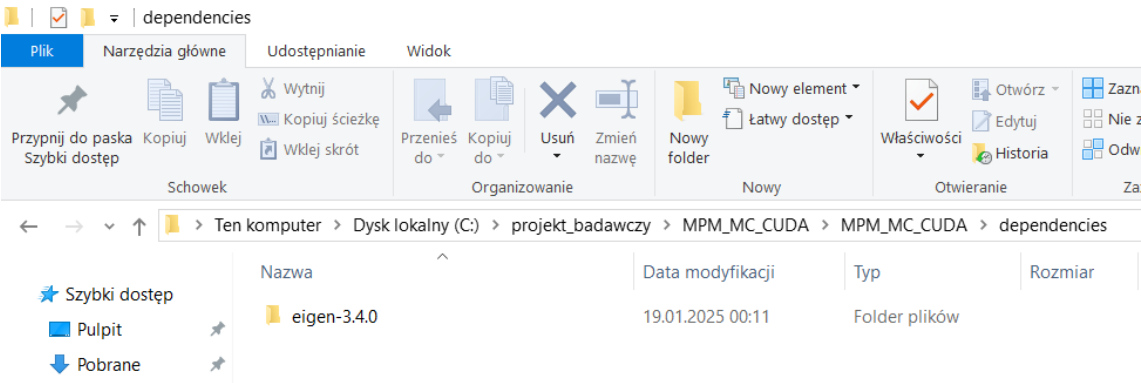
7. Teraz przygotujemy biblioteki oraz przekopiujemy zawartość katalogu src z gitlaba.

Otwieramy folder projektu (project) (ścieżka może się różnić w zależności od wybranych ścieżek w kroku 3).

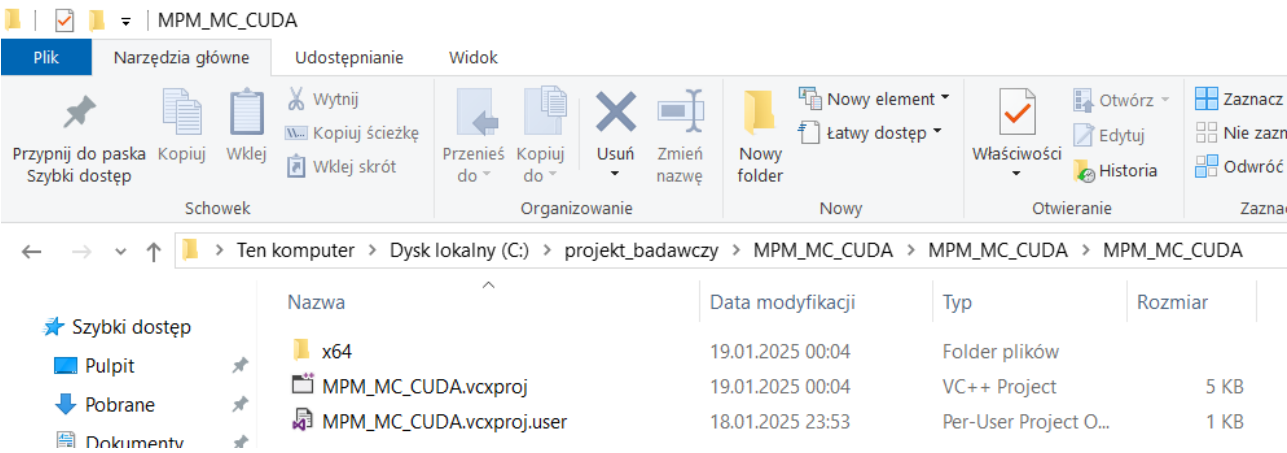


W tym folderze tworzymy katalog dependencies i kopiujemy do niego folder eigen-3.4.0 (bibliotekę należy pobrać ze strony <https://gitlab.com/libeigen/eigen/-/releases/3.4.0> lub ze strony <https://eigen.tuxfamily.org/>).

Po skopiowaniu folder dependencies powinien wyglądać następująco



Teraz będziemy kopiować kod do folderu naszego rozwiązania (solution). Obecnie powinien wyglądać jak na rysunku.



Do tego folderu należy przekopiować zawartość folderu src z gitlaba. W szczególności konieczne jest przekopiowanie plików przedstawionych na poniższym rysunku (pliki o rozszerzeniach .cu i .cuh).

| Ten komputer > Dysk lokalny (C:) > projekt_badawczy > MPM_MC_CUDA > MPM_MC_CUDA > MPM_MC_CUDA | | | |
|---|------------------|-----------------------|---------|
| Nazwa | Data modyfikacji | Typ | Rozmiar |
| x64 | 08.03.2025 22:59 | Folder plików | |
| common.cuh | 20.02.2025 20:53 | Plik CUH | 1 KB |
| emitter.cuh | 20.02.2025 20:53 | Plik CUH | 2 KB |
| Grid.cu | 20.02.2025 20:53 | Plik CU | 2 KB |
| Grid.cuh | 20.02.2025 20:53 | Plik CUH | 1 KB |
| kernel.cu | 05.03.2025 21:43 | Plik CU | 89 KB |
| MPM_MC_CUDA.vcxproj | 08.03.2025 22:59 | VC++ Project | 5 KB |
| MPM_MC_CUDA.vcxproj.user | 08.03.2025 22:58 | Per-User Project O... | 1 KB |
| Node.cuh | 20.02.2025 20:53 | Plik CUH | 1 KB |
| obj_loader.cu | 20.02.2025 20:53 | Plik CU | 6 KB |
| obj_loader.cuh | 20.02.2025 20:53 | Plik CUH | 2 KB |
| Particle.cuh | 20.02.2025 20:53 | Plik CUH | 2 KB |
| sdf.cu | 20.02.2025 20:53 | Plik CU | 7 KB |
| sdf.cuh | 20.02.2025 20:53 | Plik CUH | 3 KB |

Kolejnym potrzebnym elementem zewnętrznym jest biblioteka potrzebna do obsługi plików .json. Wykorzystamy implementację pochodzącą z repozytorium dostępnego pod linkiem: <https://github.com/nlohmann/json>. Potrzebny będzie jeden plik. Jest to json.hpp znajdujący się w wspomnianym wcześniej repozytorium w katalogu json/single_include/nlohmann/. Plik ten również umieszczamy w folderze rozwiązania.

| Ten komputer > Dysk lokalny (C:) > projekt_badawczy > MPM_MC_CUDA > MPM_MC_CUDA > MPM_MC_CUDA | | | |
|---|------------------|-----------------------|---------|
| Nazwa | Data modyfikacji | Typ | Rozmiar |
| x64 | 08.03.2025 22:59 | Folder plików | |
| common.cuh | 20.02.2025 20:53 | Plik CUH | 1 KB |
| emitter.cuh | 20.02.2025 20:53 | Plik CUH | 2 KB |
| Grid.cu | 20.02.2025 20:53 | Plik CU | 2 KB |
| Grid.cuh | 20.02.2025 20:53 | Plik CUH | 1 KB |
| json.hpp | 05.03.2025 19:06 | C/C++ Header | 956 KB |
| kernel.cu | 05.03.2025 21:43 | Plik CU | 89 KB |
| MPM_MC_CUDA.vcxproj | 08.03.2025 22:59 | VC++ Project | 5 KB |
| MPM_MC_CUDA.vcxproj.user | 08.03.2025 22:58 | Per-User Project O... | 1 KB |
| Node.cuh | 20.02.2025 20:53 | Plik CUH | 1 KB |
| obj_loader.cu | 20.02.2025 20:53 | Plik CU | 6 KB |
| obj_loader.cuh | 20.02.2025 20:53 | Plik CUH | 2 KB |
| Particle.cuh | 20.02.2025 20:53 | Plik CUH | 2 KB |
| sdf.cu | 20.02.2025 20:53 | Plik CU | 7 KB |
| sdf.cuh | 20.02.2025 20:53 | Plik CUH | 3 KB |

Następnym potrzebnym elementem jest biblioteka, która jest wykorzystywana do rozkładu macierzy na wartości osobliwe (SVD). Wykorzystujemy implementację z repozytorium <https://github.com/ericjang/svd3>. Potrzebny będzie plik svd3_cuda.h z katalogu svd3/svd3_cuda. Podobnie jak poprzednio umieszczamy go w katalogu rozwiązania.

| Ten komputer > Dysk lokalny (C:) > projekt_badawczy > MPM_MC_CUDA > MPM_MC_CUDA > MPM_MC_CUDA | | | |
|---|------------------|-----------------------|---------|
| Nazwa | Data modyfikacji | Typ | Rozmiar |
| x64 | 08.03.2025 22:59 | Folder plików | |
| common.cuh | 20.02.2025 20:53 | Plik CUH | 1 KB |
| emitter.cuh | 20.02.2025 20:53 | Plik CUH | 2 KB |
| Grid.cu | 20.02.2025 20:53 | Plik CU | 2 KB |
| Grid.cuh | 20.02.2025 20:53 | Plik CUH | 1 KB |
| json.hpp | 05.03.2025 19:06 | C/C++ Header | 956 KB |
| kernel.cu | 05.03.2025 21:43 | Plik CU | 89 KB |
| MPM_MC_CUDA.vcxproj | 08.03.2025 22:59 | VC++ Project | 5 KB |
| MPM_MC_CUDA.vcxproj.user | 08.03.2025 22:58 | Per-User Project O... | 1 KB |
| Node.cuh | 20.02.2025 20:53 | Plik CUH | 1 KB |
| obj_loader.cu | 20.02.2025 20:53 | Plik CU | 6 KB |
| obj_loader.cuh | 20.02.2025 20:53 | Plik CUH | 2 KB |
| Particle.cuh | 20.02.2025 20:53 | Plik CUH | 2 KB |
| sdf.cu | 20.02.2025 20:53 | Plik CU | 7 KB |
| sdf.cuh | 20.02.2025 20:53 | Plik CUH | 3 KB |
| svd3_cuda.h | 20.02.2025 20:53 | C/C++ Header | 15 KB |

8. Następnym krokiem jest przygotowanie katalogu, z którego zostaną pobrane siatki potrzebne do przeprowadzenia symulacji oraz katalogu, w którym zapisane zostaną wyniki.

W tym celu należy otworzyć katalog examples znajdujący się w repozytorium projektu. Tutaj rozpatrzmy przykład dla przypadku w folderze snowballs_collision. Przedstawia on symulację zderzenia niewspółosiowego dwóch kul śnieżnych.

Tworzymy nowy folder, w którym umieścimy pliki o rozszerzeniu .obj z siatkami z katalogu examples/snowballs_collision z repozytorium. Siatki te później będą wczytane do symulacji. Folder ten może znajdować się w dowolnym miejscu, tutaj przyjmiemy, że będzie to C:\projekt_badawczy\MPM_MC_CUDA\snowballs_collision. Dodatkowo w folderze tym tworzymy folder przeznaczony do przechowywania wyników symulacji, tutaj nazwiemy go „out”.

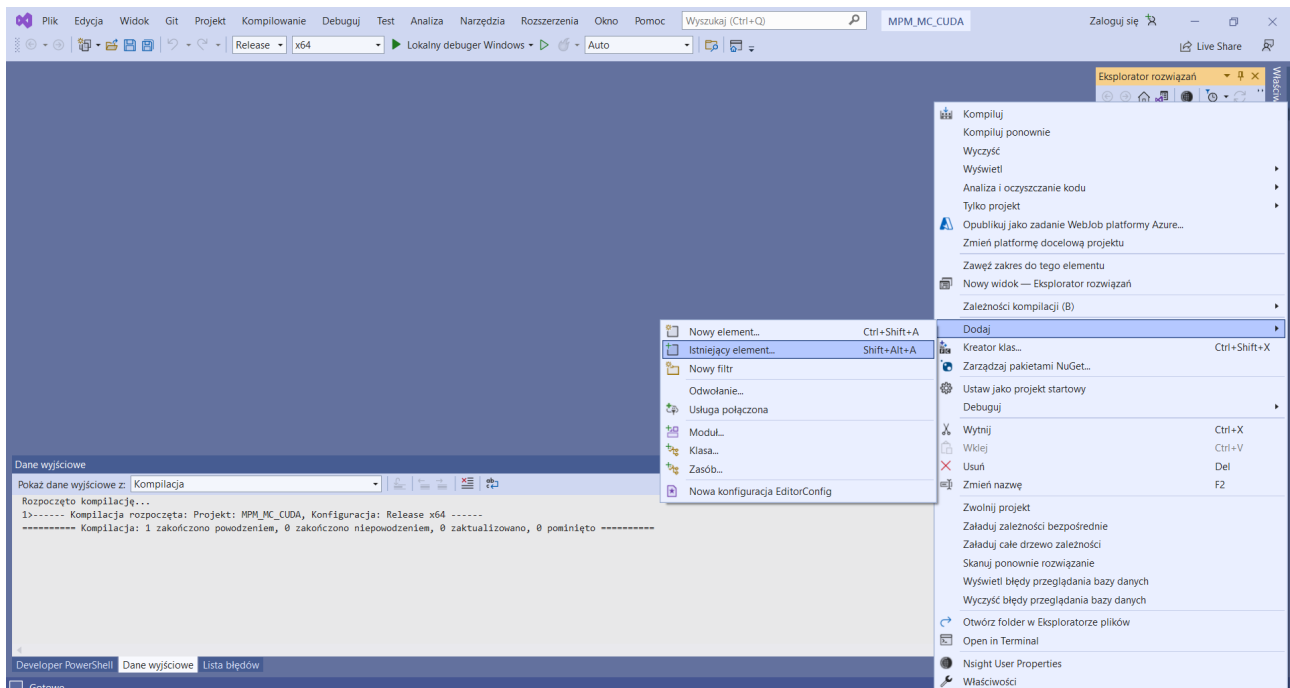
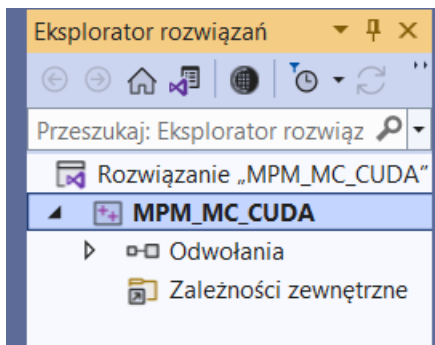
| Ten komputer > Dysk lokalny (C:) > projekt_badawczy > MPM_MC_CUDA > snowballs_collision | | | |
|---|------------------|---------------|---------|
| Nazwa | Data modyfikacji | Typ | Rozmiar |
| out | 08.03.2025 23:38 | Folder plików | |
| kule_obok_kula1.obj | 19.01.2025 01:32 | 3D Object | 7 KB |
| kule_obok_kula2.obj | 19.01.2025 01:32 | 3D Object | 7 KB |
| kule_walls.obj | 19.01.2025 01:32 | 3D Object | 1 KB |

Poza plikami .obj w katalogu examples/snowballs_collision znajduje się plik config.json. Umieszczamy go w folderze rozwiązania, w którym wcześniej umieszczaliśmy pliki z katalogu src.

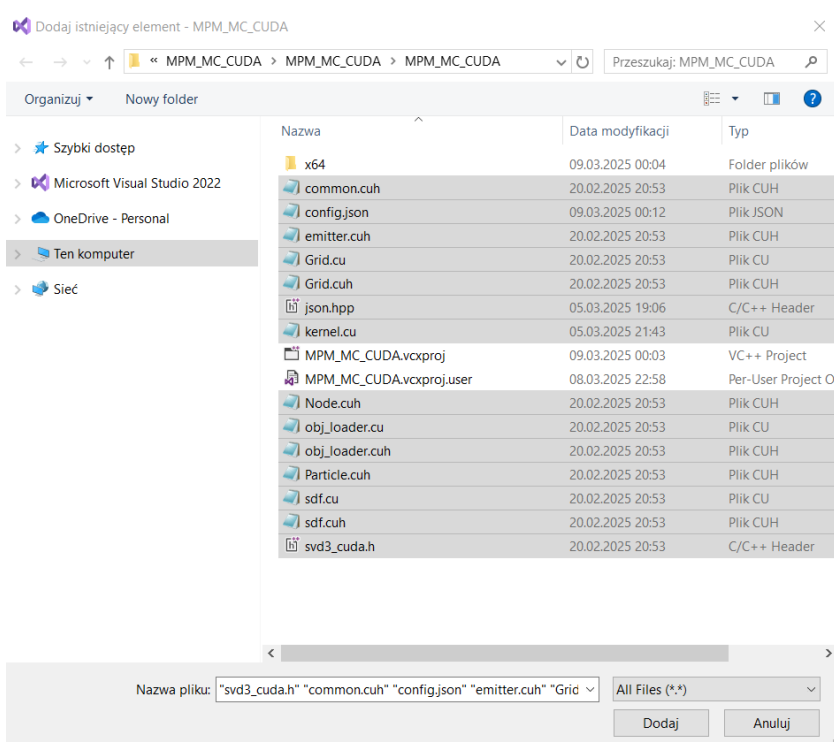
| Ten komputer > Dysk lokalny (C:) > projekt_badawczy > MPM_MC_CUDA > MPM_MC_CUDA > MPM_MC_CUDA | | | |
|---|------------------|-----------------------|---------|
| Nazwa | Data modyfikacji | Typ | Rozmiar |
| x64 | 08.03.2025 22:59 | Folder plików | |
| common.cuh | 20.02.2025 20:53 | Plik CUH | 1 KB |
| config.json | 05.03.2025 20:08 | Plik JSON | 1 KB |
| emitter.cuh | 20.02.2025 20:53 | Plik CUH | 2 KB |
| Grid.cu | 20.02.2025 20:53 | Plik CU | 2 KB |
| Grid.cuh | 20.02.2025 20:53 | Plik CUH | 1 KB |
| json.hpp | 05.03.2025 19:06 | C/C++ Header | 956 KB |
| kernel.cu | 05.03.2025 21:43 | Plik CU | 89 KB |
| MPM_MC_CUDA.vcxproj | 08.03.2025 22:59 | VC++ Project | 5 KB |
| MPM_MC_CUDA.vcxproj.user | 08.03.2025 22:58 | Per-User Project O... | 1 KB |
| Node.cuh | 20.02.2025 20:53 | Plik CUH | 1 KB |
| obj_loader.cu | 20.02.2025 20:53 | Plik CU | 6 KB |
| obj_loader.cuh | 20.02.2025 20:53 | Plik CUH | 2 KB |
| Particle.cuh | 20.02.2025 20:53 | Plik CUH | 2 KB |
| sdf.cu | 20.02.2025 20:53 | Plik CU | 7 KB |
| sdf.cuh | 20.02.2025 20:53 | Plik CUH | 3 KB |
| svd3_cuda.h | 20.02.2025 20:53 | C/C++ Header | 15 KB |

9. Kolejnym krokiem jest dodanie plików z kodem do rozwiązania w Visual Studio.

Możemy to zrobić na przykład klikając prawym przyciskiem na nazwę rozwiązania (u nas MPM_MC_CUDA). Z listy wybieramy Dodaj, a następnie Istniejący element.

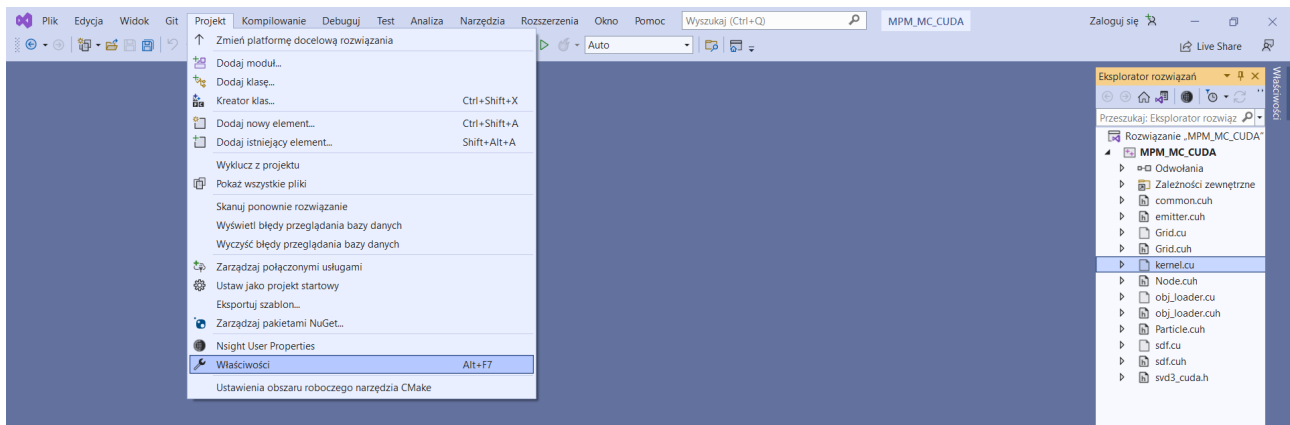


Z folderu rozwiązania wybieramy następujące pliki i zatwierdzamy przez Dodaj.

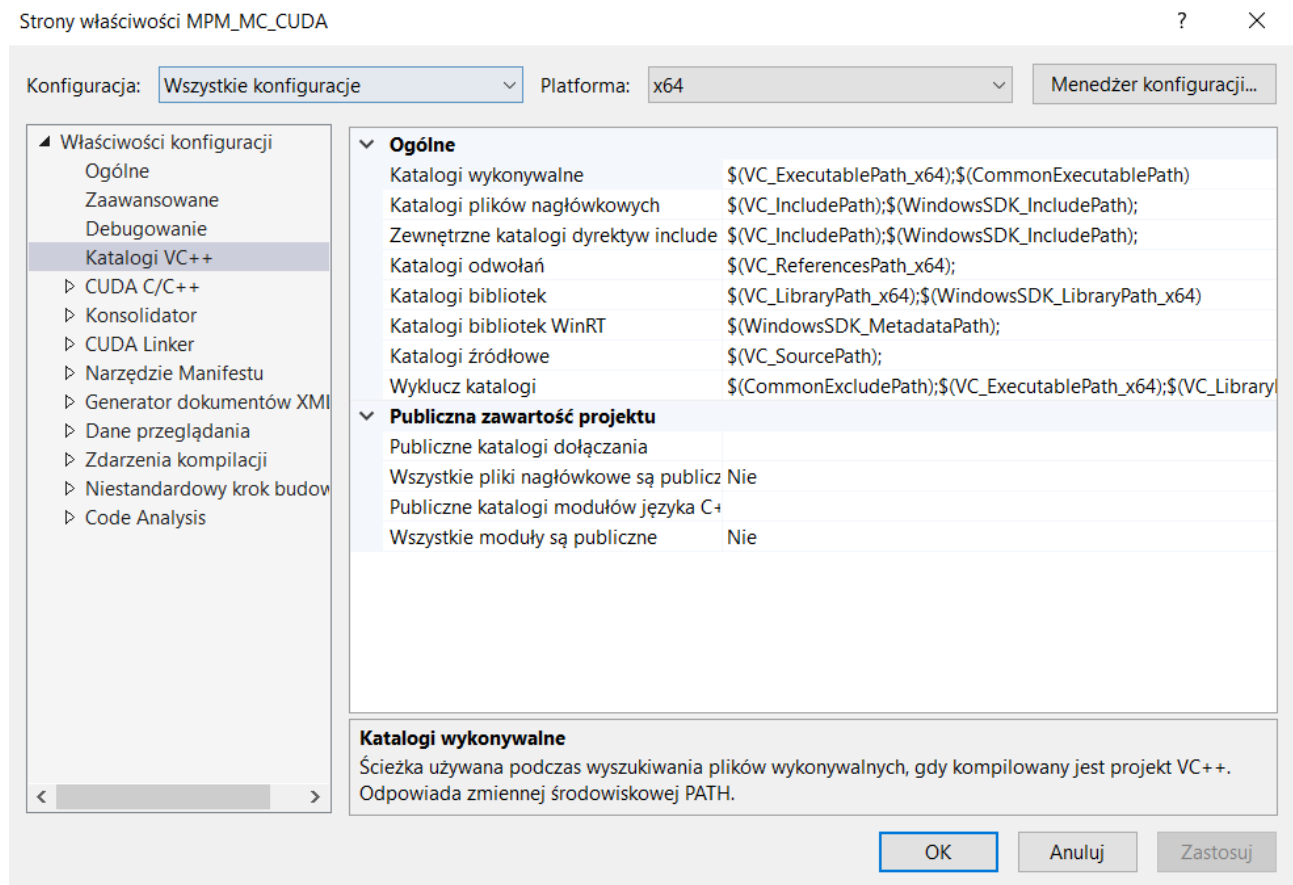


10. Następnym etapem jest dodanie biblioteki eigen do projektu.

W tym celu rozwijamy Projekt i otwieramy Właściwości

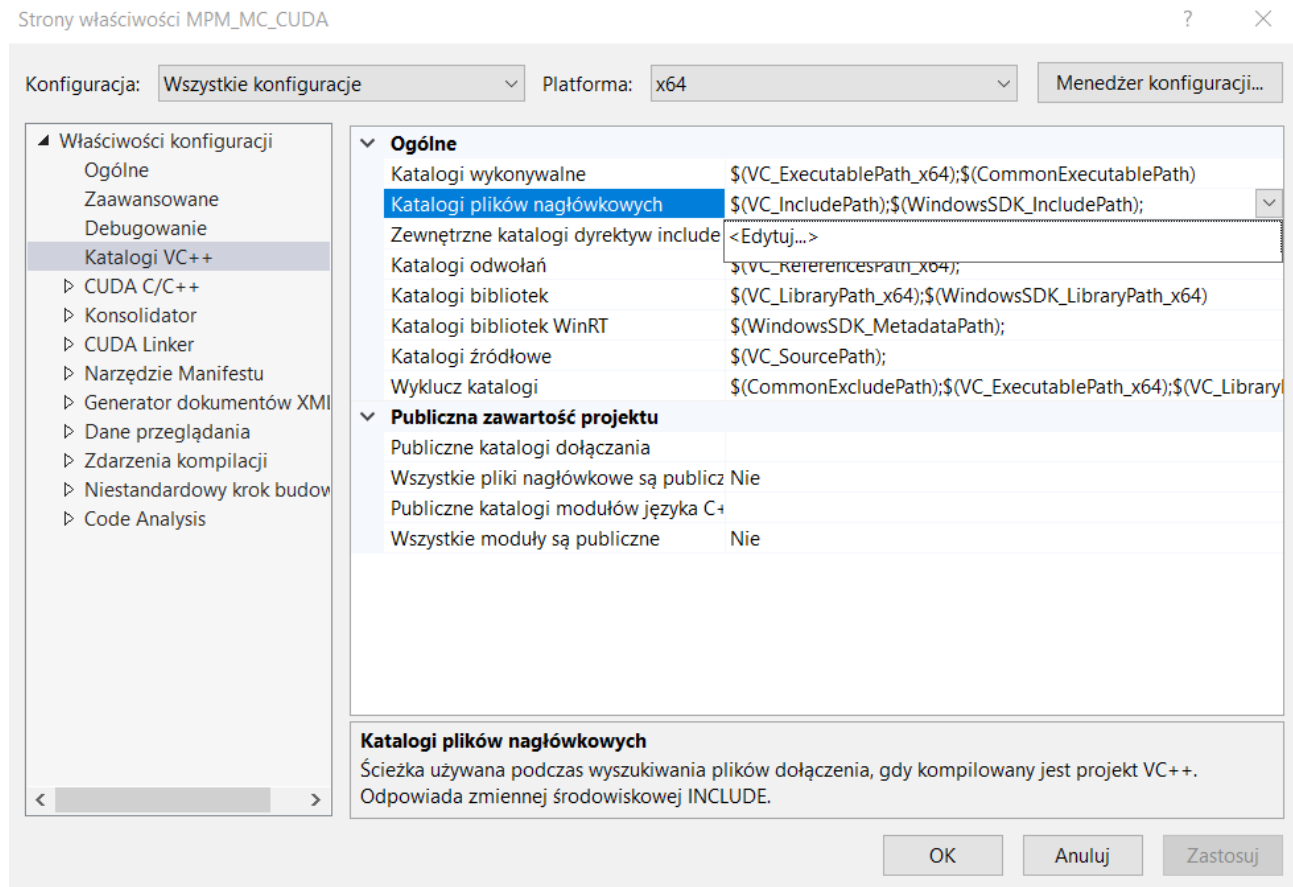


Upewniamy się, że edytujemy **Wszystkie konfiguracje**, i że platforma to x64. Wchodzimy w Katalogi VC++.

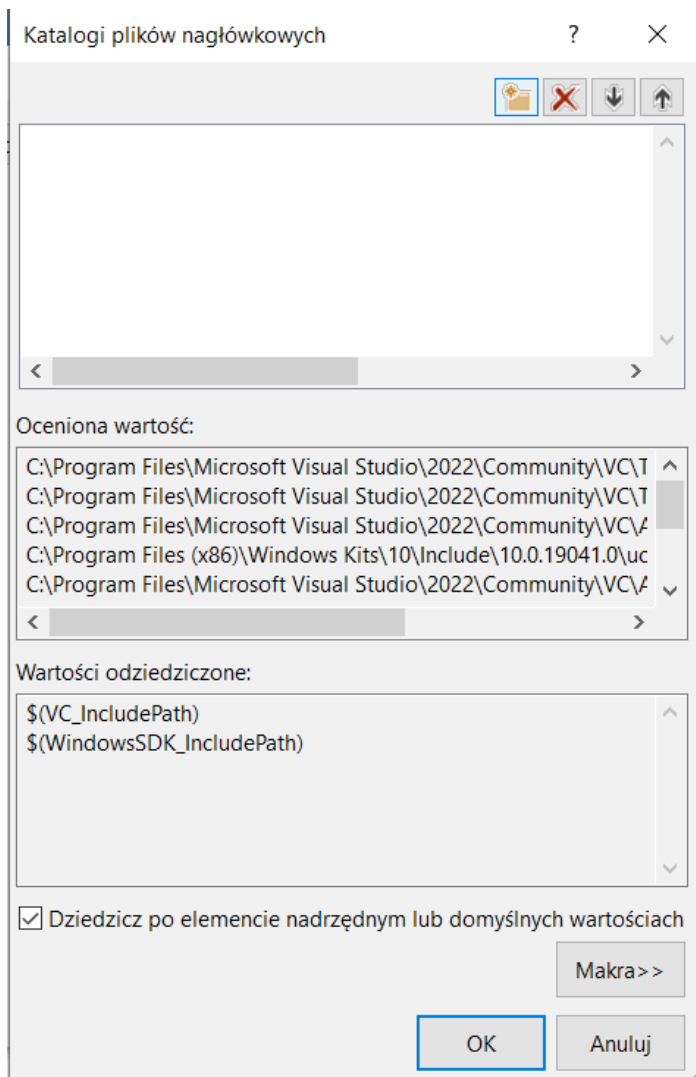


Możliwe jest, że na tym etapie część zakładek (w tym katalogi VC++) nie będzie dostępna. Należy wówczas otworzyć na przykład plik kernel.cu do edycji i ponownie wejść w właściwości projektu.

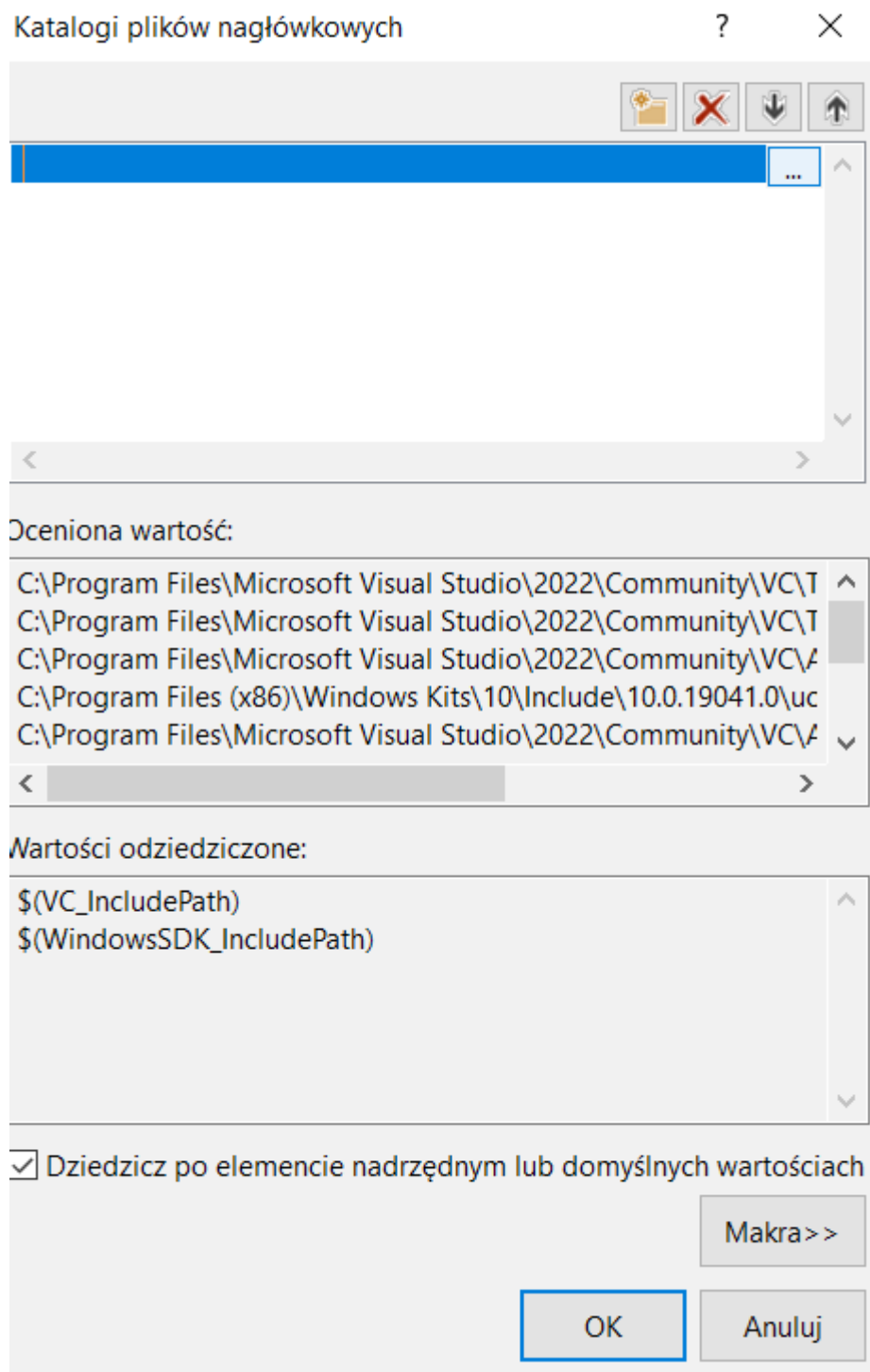
Edytujemy tutaj Katalogi plików nagłówkowych



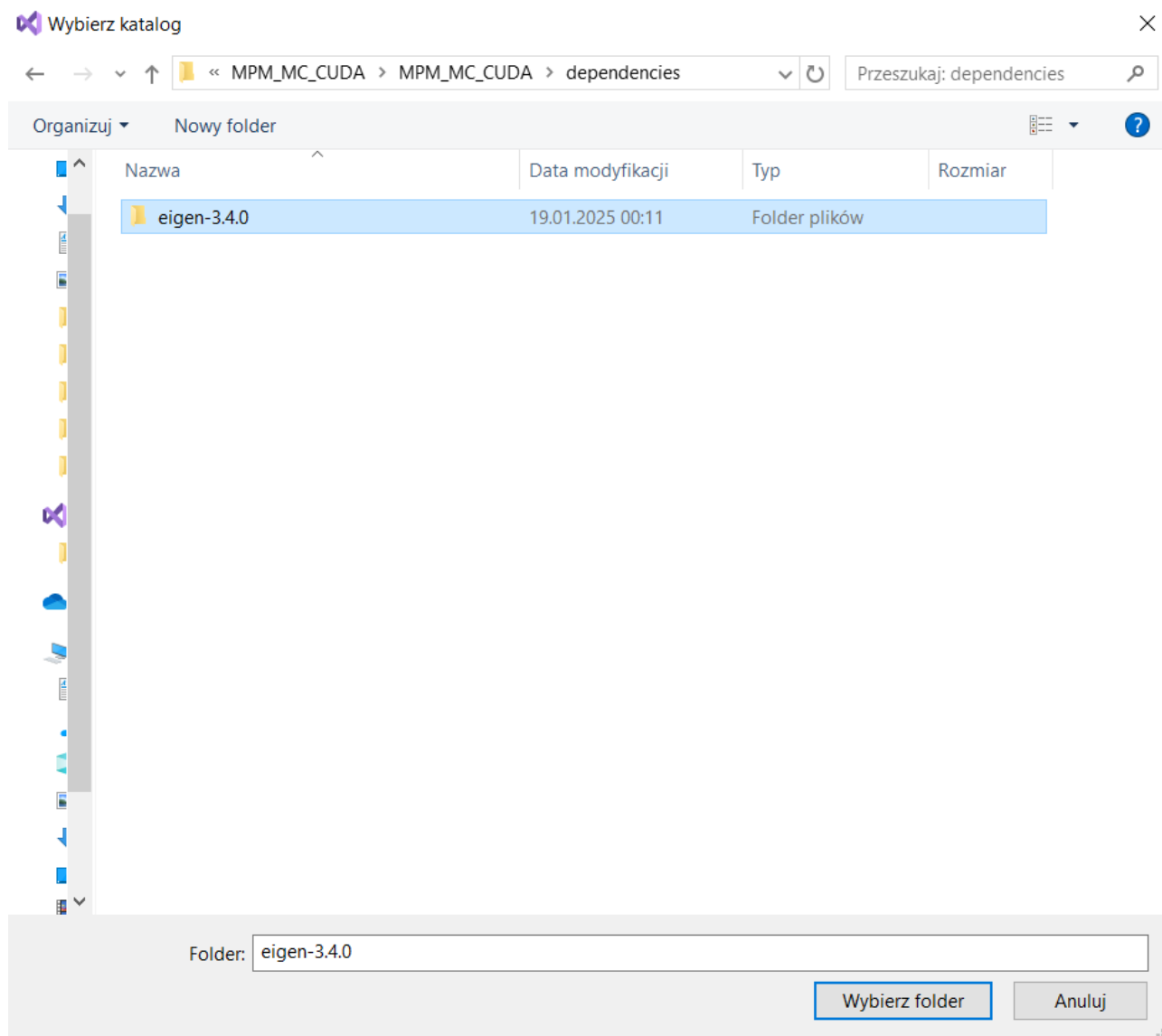
W nowym oknie dodajemy nowy katalog (przycisk na górze z niebieskim obramowaniem)



I następnie edytujemy ścieżkę przez naciśnięcie przycisku z trzema kropkami (pojawia się po najechaniu)



Przechodzimy do wcześniej przez nas utworzonego katalogu dependencies i wybieramy folder eigen, po czym zatwierdzamy przez wybierz folder. Następnie zamykamy właściwości projektu.



11. Musimy teraz dostosować plik config.json do ścieżek, które wybraliśmy do przechowywania plików siatek oraz zapisywania wyników.

W tym celu otwieramy plik config.json. Znajdujemy w nim linie pokazane na rysunku poniżej

```
28     },
29     "base_path": "C:\\ścieżka\\do\\folderu\\",
30     "paths": {
31         "result_name": "zderzenie_kul",
32         "result_path": "out\\",
33         "walls_name": "kule_walls.obj",
34         "walls_path": "",
35         "snow_objects_paths": [ "", "" ],
36         "snow_objects_names": [ "kule_obok_kula1.obj", "kule_obok_kula2.obj" ],
37         "snow_block_velocities": [
38             [ 0, 10, 0 ],
39             [ 0, -10, 0 ]
40     ],
```

Zmieniamy base_path na naszą ścieżkę, czyli C:\projekt_badawczy\MPM_MC_CUDA\snowballs_collision, co powinno wyglądać następująco

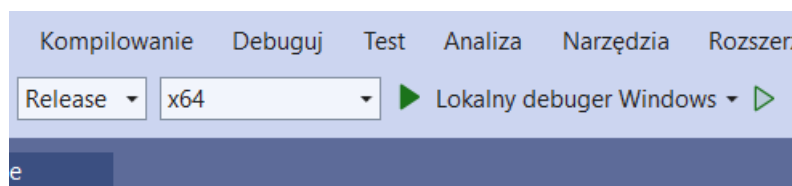
```

28     },
29     "base_path": "C:\\projekt_badawczy\\MPM_MC_CUDA\\snowballs_collision\\",
30     "paths": {
31         "result_name": "zderzenie_kul",
32         "result_path": "out\\",
33         "walls_name": "kule_walls.obj",
34         "walls_path": "",
35         "snow_objects_paths": [ "", "" ],
36         "snow_objects_names": [ "kule_obok_kula1.obj", "kule_obok_kula2.obj" ],
37         "snow_block_velocities": [
38             [ 0, 10, 0 ],
39             [ 0, -10, 0 ]

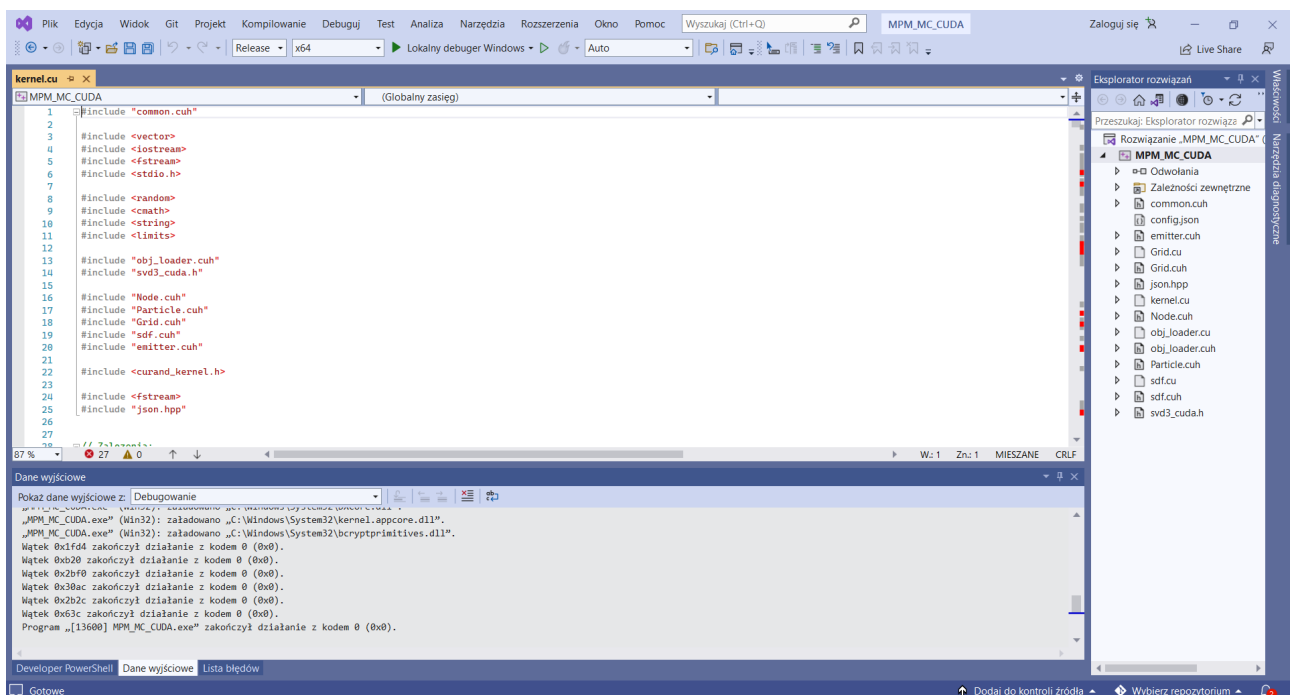
```

W przypadku nazwania folderu do przechowywania wyników inaczej niż out, należy zmienić również odpowiednio result_path.

12. W celu zwiększenia szybkości działania symulacji możemy zmienić konfigurację rozwiązania z Debug na Release.



13. Projekt powinien być możliwy do skompilowania. Przy otwartym jedynie pliku kernel.cu liczba błędów wynosi 27.



14. Wynik w konsoli powinien wyglądać początkowo











```
WybierzKonsola debugowania programu Microsoft Visual Studio
Theta_c: 0.019
Theta_s: 0.0075
xi: 10
rho_0: 100
Modul Younga (E_0): 150000
nu: 0.2
Wsp~czynnik Lamego (mu_0): 62500
Wsp~czynnik Lamego (lambda_0): 41666.7
Krok czasowy (dt): 0.00025
t: 0
t_0: 0
t_end: 50
sticky: 0
alpha: 0.95
particle_diameter: 0.0144
h: 0.03125
MARCHING_CUBES_EPSILON: 1e-06
isolevel: 0.001
step: 10
base_path: C:\projekt_badawczy\MPM_MC_CUDA\snowballs_collision\
result_path: C:\projekt_badawczy\MPM_MC_CUDA\snowballs_collision\out\
result_name: zderzenie_kul
walls_path: C:\projekt_badawczy\MPM_MC_CUDA\snowballs_collision\
walls_name: kule_walls.obj
n: 88713
n: 88713
C:\projekt_badawczy\MPM_MC_CUDA\snowballs_collision\out\
=====
Timestep number: 0
Simulation time: 0.00025
```

a później

```
WybierzKonsola debugowania programu Microsoft Visual Studio
Simulation time: 0.0005
Total mass in simulation: 52.9463
=====
Timestep number: 2
Simulation time: 0.00075
Total mass in simulation: 52.9463
=====
Timestep number: 3
Simulation time: 0.001
Total mass in simulation: 52.9463
=====
Timestep number: 4
Simulation time: 0.00125
Total mass in simulation: 52.9463
=====
Timestep number: 5
Simulation time: 0.0015
Total mass in simulation: 52.9463
=====
Timestep number: 6
Simulation time: 0.00175
Total mass in simulation: 52.9463
=====
Timestep number: 7
Simulation time: 0.002
Total mass in simulation: 52.9463
=====
Timestep number: 8
Simulation time: 0.00225
Total mass in simulation: 52.9463
```

Wyniki w postaci plików .obj pojawiają się w folderze out.

Ten komputer > Dysk lokalny (C:) > projekt_badawczy > MPM_MC_CUDA > snowballs_collision > out

| Nazwa | Data modyfikacji | Typ | Rozmiar |
|--|------------------|-----------|----------|
|  zderzenie_kul0.obj | 09.03.2025 00:17 | 3D Object | 1 432 KB |
|  zderzenie_kul1.obj | 09.03.2025 00:17 | 3D Object | 1 434 KB |
|  zderzenie_kul2.obj | 09.03.2025 00:17 | 3D Object | 1 432 KB |
|  zderzenie_kul3.obj | 09.03.2025 00:17 | 3D Object | 1 439 KB |
|  zderzenie_kul4.obj | 09.03.2025 00:17 | 3D Object | 1 440 KB |
|  zderzenie_kul5.obj | 09.03.2025 00:17 | 3D Object | 1 437 KB |
|  zderzenie_kul6.obj | 09.03.2025 00:17 | 3D Object | 1 436 KB |
|  zderzenie_kul7.obj | 09.03.2025 00:17 | 3D Object | 1 439 KB |
|  zderzenie_kul8.obj | 09.03.2025 00:17 | 3D Object | 1 445 KB |
|  zderzenie_kul9.obj | 09.03.2025 00:17 | 3D Object | 1 447 KB |