PROJEKT. Symulacja Układu Słonecznego.

Mateusz Szmyd 179920

Karta graficzna: Geforce GTX 1050Ti Mobile.

Do graficznego przedstawienia symulacji została wykorzystana biblioteka VPython języka Pytohn. Za pomocą dynamicznej biblioteki dla linuxa (shared object file .so) importowane są funkcje z języka C++ razem z obsługą karty graficznej (CUDA).

W celu zachowania kompatybilności pomiędzy dwoma językami, w Pythonie zostały napisane klasy dziedziczące po ctypes.Structure, które odzwierciedlają struktury/klasy napisane w języku CUDA/C++.

Obliczenia wykonywane są z dokładnością do jednej minuty (dt = 60s), dla większych wartości np. dla godziny. orbity planet powiększają się w niewielkim stopniu po każdym okrążeniu Słońca.

Większość danych, w tym pozycje i prędkości obiektów, zostały ręcznie skopiowane z aplikacji https://ssd.jpl.nasa.gov/horizons/app.html#/. Brakujące dane dla małych obiektów (masa, gęstość oraz promień) zostały pozyskane z: https://en.wikipedia.org/wiki/List_of_Solar_System_objects_by_size oraz z podstron zawierających szczegółowe dane o konkretnych ciałach niebieskich.

29.12.2022	POSX [km]	POSY [km]	POSZ [km]	NASA POSX [km]	NASA POSY [km]	NASA POSZ [km]
Sun	-1.35522E+06	1.90795E+04	3.14111E+04	-1.35505E+06	1.82462E+04	3.14159E+04
Mercury	3.36819E+07	3.21836E+07	-5.54078E+05	3.06242E+07	3.45928E+07	-7.64535E+04
Venus	7.64776E+07	-7.60615E+07	-5.50433E+06	7.65967E+07	-7.58785E+07	-5.50856E+06
Earth	-1.90524E+07	1.46090E+08	2.40805E+04	-1.90635E+07	1.46067E+08	2.42468E+04
Mars	1.37349E+07	2.32956E+08	4.54328E+06	1.37860E+07	2.32943E+08	4.54169E+06
Jupiter	7.22938E+08	1.52545E+08	-1.66525E+07	7.23181E+08	1.52909E+08	-1.68139E+07
Saturn	1.22053E+09	-8.28022E+08	-3.41186E+07	1.21621E+09	-8.25608E+08	-3.40677E+07
Uranus	2.01924E+09	2.15024E+09	-1.83140E+07	2.00004E+09	2.15715E+09	-1.78991E+07
Neptune	4.43161E+09	-4.44661E+08	-9.37152E+07	4.45068E+09	-4.41635E+08	-9.34758E+07
Pluto	2.39490E+09	-4.62464E+09	-2.09420E+08	2.41738E+09	-4.58420E+09	-2.08711E+08
Moon	-1.92016E+07	1.45773E+08	1.24978E+04	-1.86925E+07	1.46026E+08	-4.77445E+02

Powyżej znajduje się tabela dla pozycji ciał niebieskich z symulacji oraz z bazy danych NASA po roku symulacji.

29.12.2022	Błąd POSX [%]	Błąd POSY [%]	Błąd POSZ [%]
Sun	0.01%	4.57%	0.02%
Mercury	9.98%	6.96%	624.73%
Venus	0.16%	0.24%	0.08%
Earth	0.06%	0.02%	0.69%
Mars	0.37%	0.01%	0.03%
Jupiter	0.03%	0.24%	0.96%
Saturn	0.36%	0.29%	0.15%
Uranus	0.96%	0.32%	2.32%
Neptune	0.43%	0.69%	0.26%
Pluto	0.93%	0.88%	0.34%
Moon	2.72%	0.17%	2717.64%

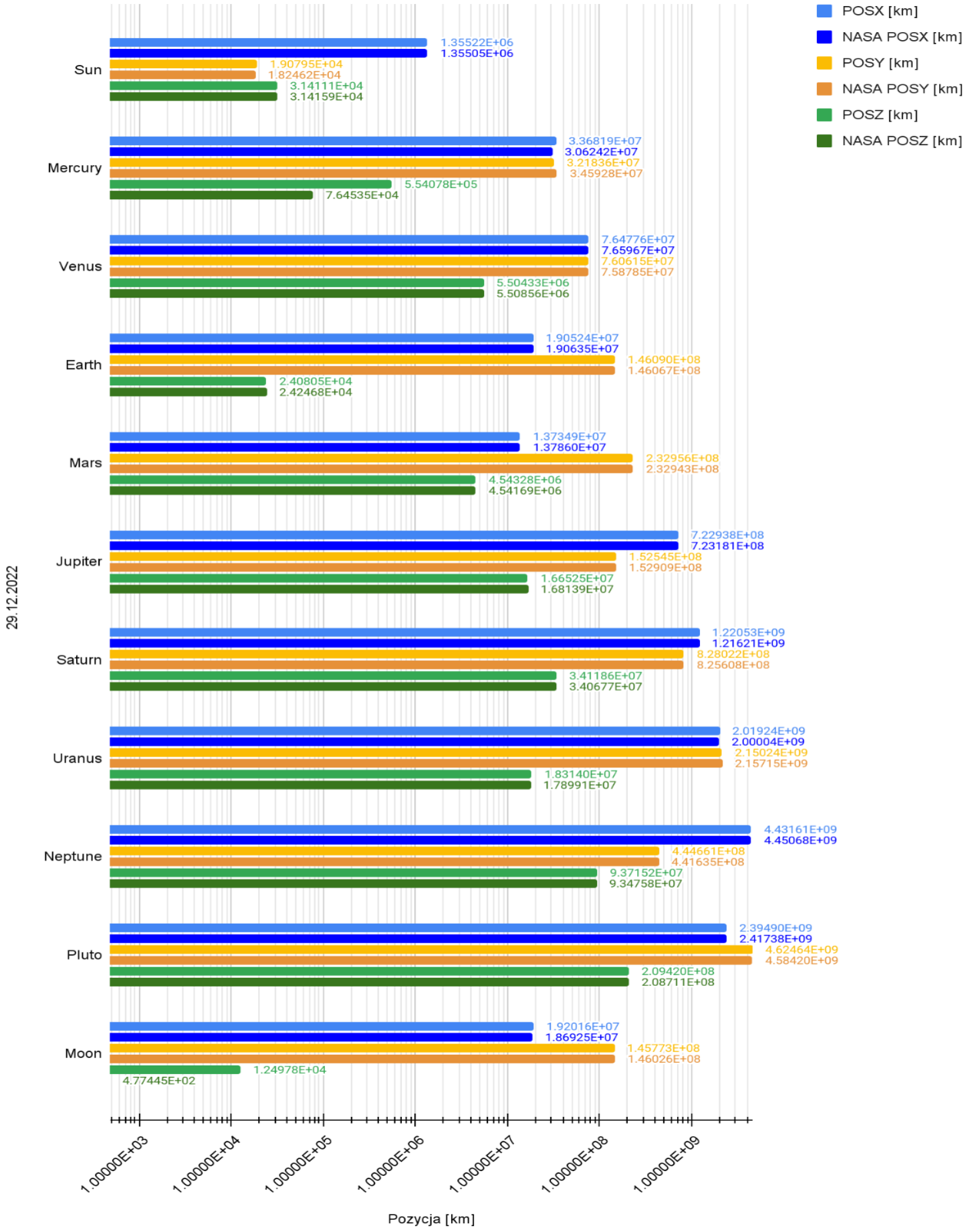
Powyżej znajduje się tabela z błędami względnymi dla pozycji obiektów.

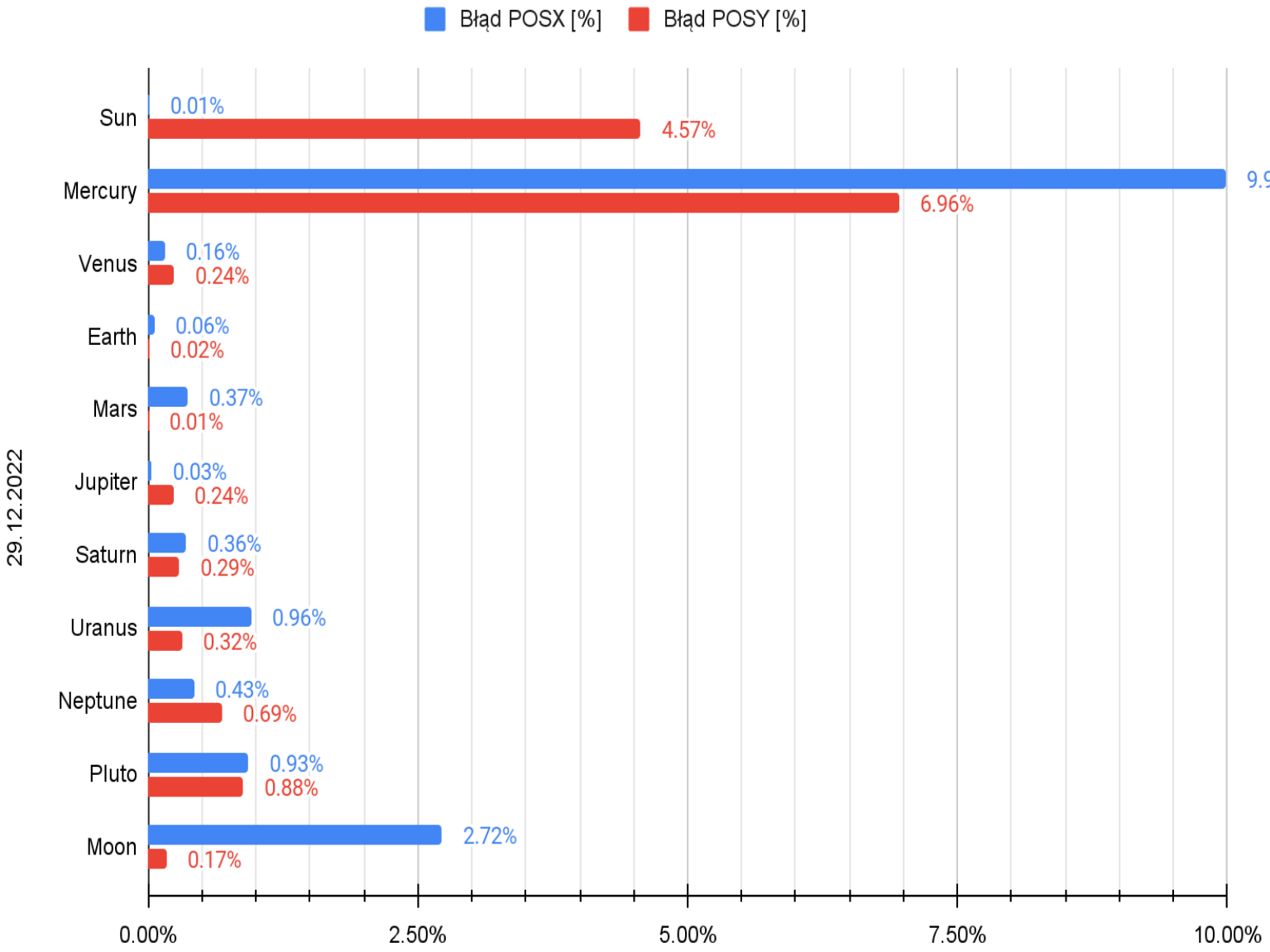
Można zauważyć, że największym błędem obarczone są pozycje (szczególnie Y) dla małych obiektów (o niewielkiej masie). Błędy wynikają z innej dokładności wykonywanych symulacji. Również wykonywana symulacja posiada model, który nie bierze pod uwagę wielu innych obiektów.

Link do arkuszy Google zawierających pomiary oraz wykresy: Projekt

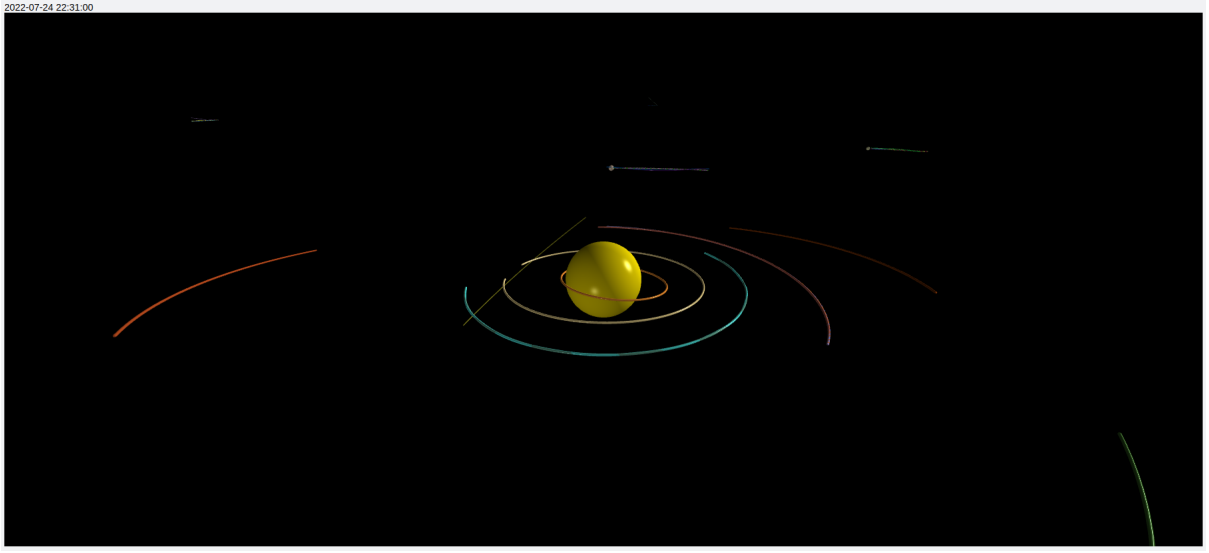
Poniżej znajdują się dane z tabel w formie wykresów:

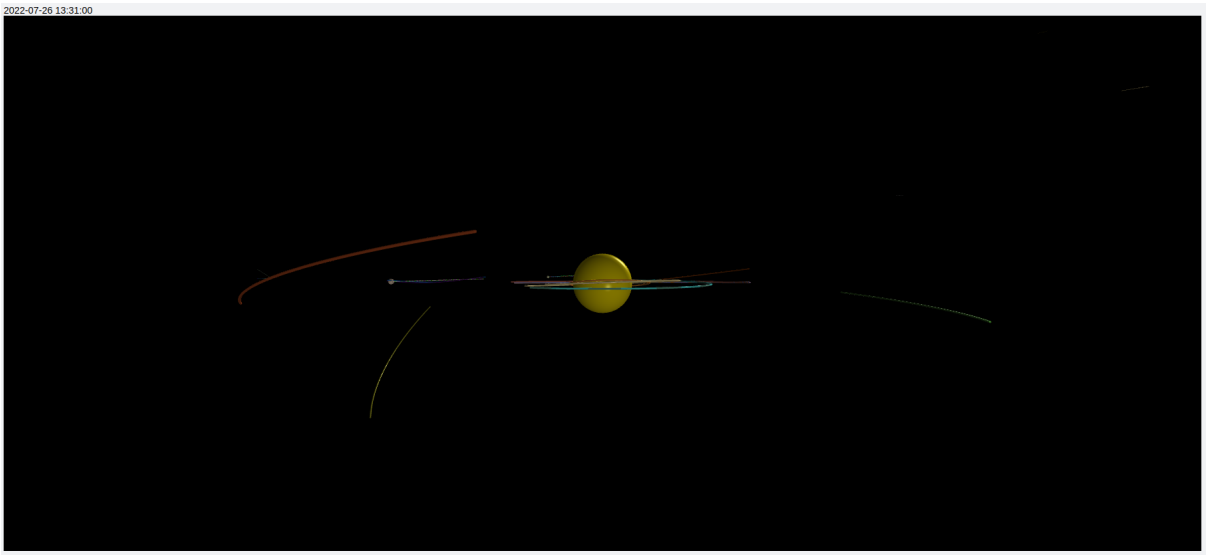
Pozycje x, y, z głównych obiektów Układu Słonecznego po roku symulacji

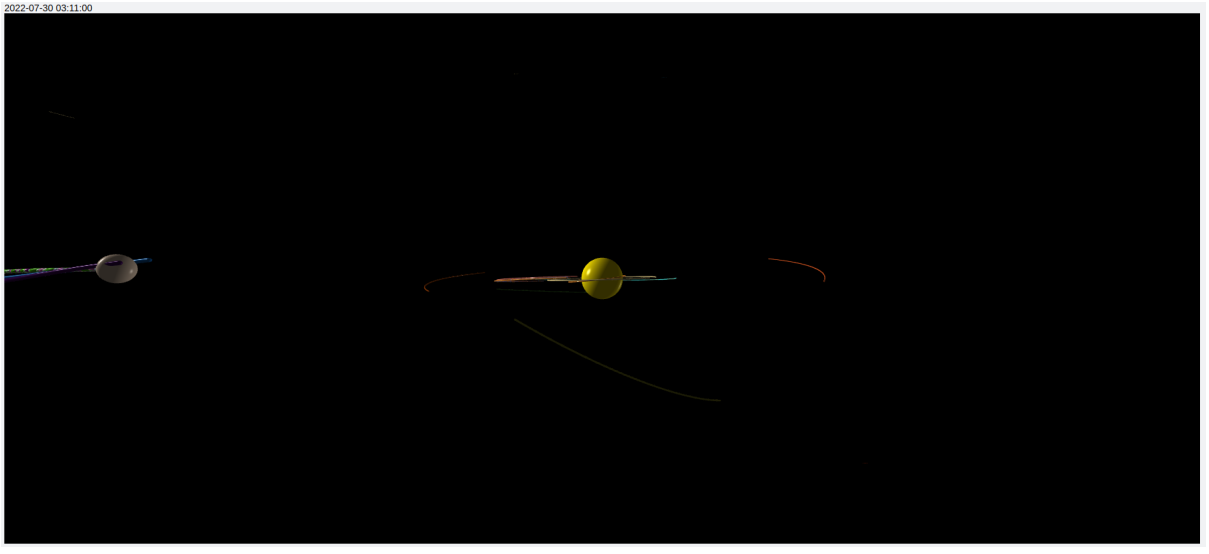


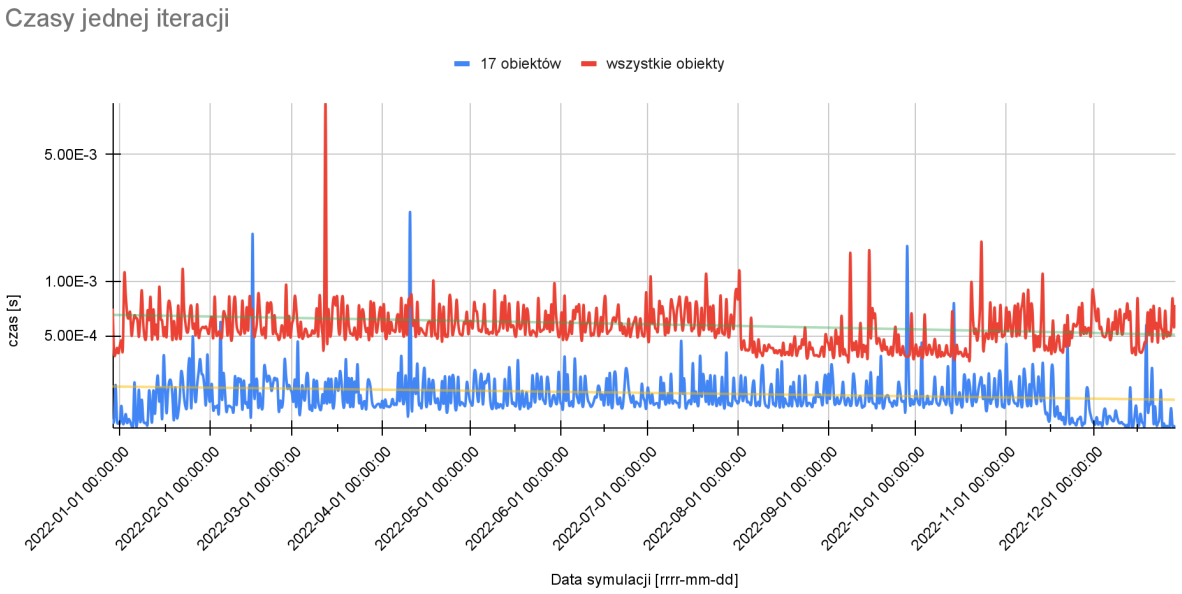

Błąd POSX [%]
Błąd POSY [%]
29.12.2022
Sun
0.01%
4.57%
Mercury
9.9
6.96%
Venus
0.16%
0.24%
Earth
0.06%
0.02%
Mars
0.37%
0.01%
Jupiter
0.03%
0.24%
Saturn
0.36%
0.29%
Uranus
0.96%
0.32%
Neptune
0.43%
0.69%
Pluto
0.93%
0.88%
Moon
2.72%
0.17%
0.00%
2.50%
5.00%
7.50%
10.00%

Zrzuty symulacji z różnych perspektyw:



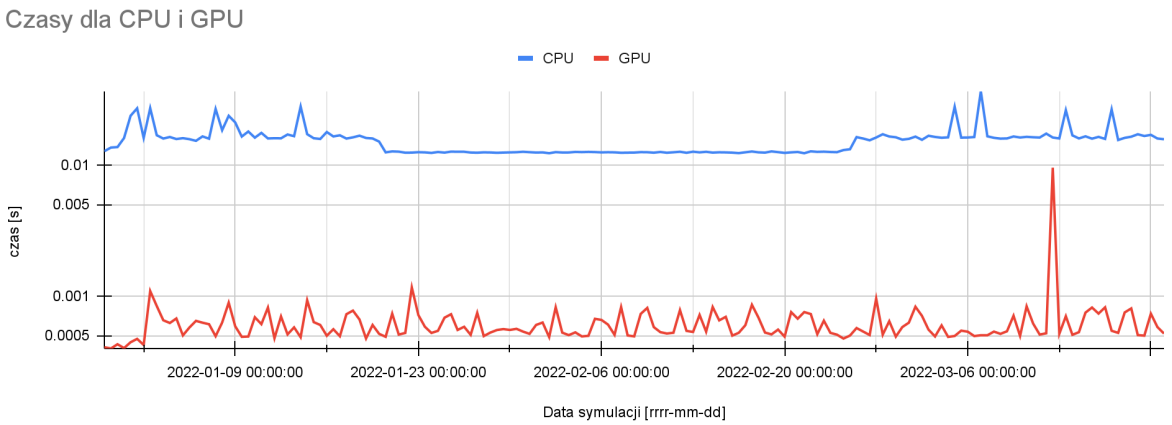






Powyżej znajduje się wykres przedstawiający czasy dla jednej pętli programu. Można zauważyć, że 3,3-krotne zwiększenie liczby obiektów spowodowało 2,5-krotny wzrost czasów.

Kolejny wykres dotyczy porównania czasów na CPU i GPU dla wszystkich obiektów.



Czas trwania jednej pętli na CPU jest większy średnio 26,6-krotnie.

Największym wyzwaniem w projekcie było efektywne połączenie języka C++/CUDA z Pythonem. Wynikało z tego wiele problemów, takich jak:

- brak kompatybilności pomiędzy typami zmiennych,
- różnice pomiędzy listami pythonowskimi, a tablicami C++,
- przechowywaniem zmiennych:

  Nie była możliwa inicjalizacja pamięci oraz kopiowanie danych do device'a za pomocą jednej funkcji wykonanej jednokrotnie na początku programu. Symulacja wykonywała się i był ruch obiektów, ale nie utrzymywały swojej orbity. Ostatecznie wykonywana jest inicjalizacja, kopiowanie i uwalnianie pamięci na device w każdej iteracji. Jest to główny powód zmniejszenia częstotliwości wykonywanych obliczeń.

Wniosek

Nawet przy tak niewielkiej ilości obiektów karta graficzna osiąga lepsze czasy niż CPU.