Politechnika Śląska

Politechnika Śląska
Wydział Automatyki Elektroniki i Informatyki

Języki Asemblerowe

Kamil Kowalski (295672)

10 stycznia 2023

Politechnika Śląska

Table of Contents

1 Opis założeń projektu

► Opis założeń projektu

► Porównanie wydajności

Temat projektu

1 Opis założeń projektu

- Tematem prezentowanego projektu jest program do generacji siatki torusa.
- Wynikiem działania programu będzie tablica zawierająca uporządkowane wierzchołki bryły.
- Wygenerowana siatka będzie zawierać 2 typy danych: pozycje oraz wektory normalne.
- Program powinien wykorzystywać wielowątkowość do przyśpieszenia obliczeń

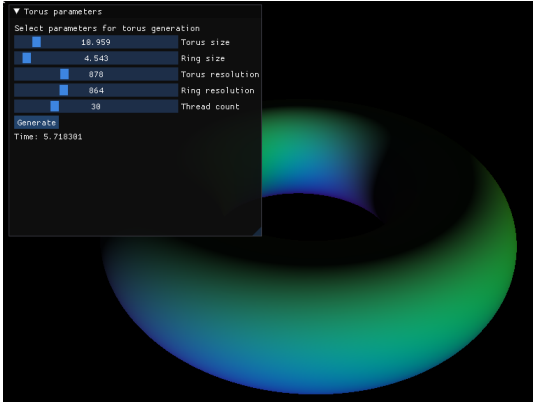
Działanie projektu

1 Opis założeń projektu

Program będzie wyświetlał interface użytkownika oraz wygenerowaną bryłę w scenie z ruchomym źródłem światła. Użytkownik będzie w stanie dostosować następujące parametry:

- Rozdzielczość pierścienia
- Rozdzielczość torusa
- Rozmiar pierścienia
- Rozmiar torusa
- Ilość wątków roboczych

Efekt końcowy



Podział na biblioteki

Program będzie składał się z 3 części

- Aplikacja główna w C++ oraz OpenGL wykorzystująca ImGui do UI
- Biblioteka dynamiczna do generacji torusa w języku C++
- Biblioteka dynamiczna do generacji torusa w dialekcie NASM języka Asembler

Taki podział programu umożliwi późniejsze porównanie wydajności obu bibliotek

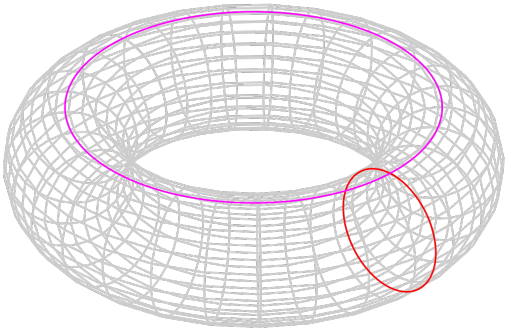
Algorytm generacji

Algorytm dzieli torus na pierścienie, następnie wierzchołki pierścienia są wyznaczane według wzoru:

$$\begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} \cos\alpha\,(a + b\cos\beta) \\ b\sin\beta \\ \sin\alpha\,(a + b\cos\beta) \end{bmatrix} \qquad \begin{bmatrix} N_x \\ N_y \\ N_z \end{bmatrix} = \begin{bmatrix} \cos\alpha\,(\cos\beta) \\ \sin\beta \\ \sin\alpha\,(\cos\beta) \end{bmatrix}$$

Politechnika Śląska

Algorytm generacji - przykład



Rysunek: Ilustracja działania algorytmu [1]

[1] wikipedia

Algorytm generacji - wielowątkowość

Ponieważ każdy pierścień siatki może być obliczany niezależnie od innych, zadanie wyznaczenie siatki pojedynczego pierścienia nadaje się do rozdzielenia pomiędzy wątki, których liczba nie może być mniejsza niż rozdzielczość siatki torusa.
Obecna implementacja dla n wątków i k pierścieni każdemu wątkowi przydziela pomiędzy $\lfloor \frac{k}{n} \rfloor$ a $\lceil \frac{k}{n} \rceil$ kolejnych pierścieni do wyznaczenia, co umożliwia równomierne obciążenie wątków.

Politechnika Śląska

Table of Contents

2 Porównanie wydajności

▶ Opis założeń projektu

▶ Porównanie wydajności

Metodyka badania

2 Porównanie wydajności

Następujące testy zostały przeprowadzone na komputerze z systemem operacyjnym Linux w wersji jądra 5.15.86 z procesorem AMD Ryzen 5 1500X 4 rdzenie, 8 wątków.
Testy były wykonane na maksymalnie odciążonym systemie (z wyłączoną większością procesów w tle)
Program C++ został skompilowany z użyciem kompilatora `gcc` z flagami `-mavx -O3`
Program assemblerowy został zasemblowany i zlinkowany w następujący sposób:

```
nasm -felf64 torusgen.asm && ld -shared torusgen.o -lm -o libgentorus.so
```

C++ vs. Asembler

2 Porównanie wydajności

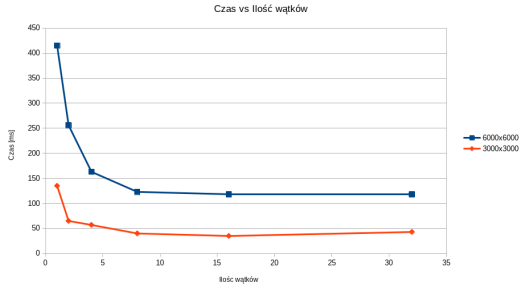
Wyniki dla wykonania na pojedynczym rdzeniu

t [ms]	2000x2000	3000x3000	4000x4000
C++	66.53	48.01	21.59
ASM	68.13	43.43	20.77

Politechnika Śląska

Czas działania względem ilości wątków (ASM)

2 Porównanie wydajności



t [ms]	1	2	4	8	16	32
6000x6000	415	256	163	123	118	118
3000x3000	135	65	57	40	35	43

Politechnika Śląska

Dziękuje za uwagę