Procesory graficzne w obliczeniach równoległych (CUDA)

Projekt

Wersja (beta): 9 grudnia 2010

Projekt końcowy składa się z dwóch niezależnych zadań **A** i **B**. Oba zadania należy oddać przed końcem semestru, a jedno zadanie zaprezentować przed 13 stycznia.

Część A: Otoczka wypukła

Zadanie polega na zaimplementowaniu algorytmu obliczającego otoczkę wypukłą zbioru punktów 2D. Czyli trzeba znaleść podzbiór danego zbioru punktów wyznaczający wielokąt wypukły będący najmniejszym zbiorem wypukłym zawierającym wszystkie dane. Dokładniejsze informacje o możliwej metodzie implementacji i algorytmie w pracy Srikanth'a poniżej.

Pierwszym argumentem programu testującego ma być napis `cpu` lub `gpu` określający metodę. Kolejne argumenty oznaczają ilości losowanych punktów w tysiącach. Punkty losujemy jednorodnie z koła o promieniu 1 na płaszczyźnie XY. Program powinien iterując po argumentach dla każdego kolejno losować punkty, wyświetlać na ekranie, obliczać otoczkę wypukłą, wypisywać czas działania (bez przesyłania danych pomiędzy cpu i gpu), wyświetlać punkty otoczki innym kolorem lub wielokąt na nich bazujący. Po skończeniu iteracji program powinien zakończyć działanie co umożliwi z zewnątrz zmierzenie całkowitego czasu działania bez interakcji.

Punkty wyświetlamy przy pomocy OpenGL z domyślnym rzutem równoległym, który wyświetla obszar $[-1, 1] \times [-1, 1]$. Wylosowane punkty umieszczamy w VBO (Vertex Buffer Object) tak jak np. w przykładzie `simpleGL` z SDK lub analogicznie jak w zadaniu 5 wykorzystywaliśmy PBO (Pixel Buffer Object).

Najszybsze programy będą dodatkowo punktowane.

Literatura:

1. Srikanth, D., Kothapalli, K., Govindarajulu, R., and Narayanan, "Parallelizing Two Dimensional Convex Hull on NVIDIA GPU and Cell BE", 2009.
   `http://www.hipc.org/hipc2009/documents/HIPCSS09Papers/1569255641.pdf`
2. Tomasz Jurkiewicz, Piotr Danilewski, "Efficient Quicksort and 2D Convex Hull for CUDA, and MPRAM as a Realistic Model of Massively Parallel Computations", Submitted for 1st International ICST Conference on Theory and Practice of Algorithms in (Computer) Systems - TAPAS 2011.
   `http://www.mpi-inf.mpg.de/~tojot/papers/chull.pdf`

Część B: Ray tracing powierzchni implicit

Powierzchnią implicit stopnia n w przestrzeni nazywamy zbiór punktów spełniających równanie $F(x, y, z) = 0$ gdzie F jest wielomianem stopnia n. Metoda ray tracingu polega na poprowadzeniu dla każdego piksela ekranu promienia od obserwatora i znalezieniu przecięcia z powierzchnią. Jeśli mamy promień $R(t) = [X(t), Y(t), Z(t)]^T$ to aby znaleść przecięcie wystarczy wstawić współrzędne do równania opisującego powierzchnie uwikłaną aby otrzymać równanie jednej zmiennej $F(X(t), Y(t), Z(t)) = 0$, które jest wielomianem stopnia n i którego rozwiązania t odpowiadają punktom przecięcia $R(t)$.

Dla zaznaczenia kształtu powierzchni kolor wyznaczamy na postawie współrzędnych XYZ, można też zaimplementować proste oświetlenie z punktowym lub kierunkowym źródłem światła. W każdym punkcie przecięcia trzeba wtedy policzyć wektor normalny, który jest gradientem funkcji F (wektor pochodnych cząstkowych).

Program powinien służyć eksploracji powierzchni i w sposób interakcyjny umożliwiać modyfikacje równania dla którego wyświetlamy powierzchnię implicit. Minimum to zaimplementowanie równań stopnia 2 i 3 dla których są gotowe wzory na pierwiastki. Program powinien umożliwiać wybór spośród powierzchni, które są wymienione jako przykłady w poniższej pracy (dla stopnia 3 mamy tam Cayley, Clebsch, Ding-Dong). Można też dodać własne powierzchnie i własne ciekawe widoki.

Informacje o szybkości nalezy podawać w postaci FPS (frames per second).

Literatura:

1. Mohan Singh, P.J. Narayanan, "Real-Time Ray Tracing of Implicit Surfaces on the GPU" IEEE Transactions on Visualization and Computer Graphics, 27 Mar. 2009. IEEE computer Society Digital Library.
`http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.99.9031&rep=rep1&type=pdf`

Andrzej Łukaszewski