AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Wydział Informatyki, Elektroniki i Telekomunikacji
Katedra Informatyki

PROJEKT INŻYNIERSKI

IMPLEMENTACJA METODY SPH NA PROCESORY GRAFICZNE

DAWID ROMANOWSKI, WOJCIECH CZARNY

OPIEKUN:
dr hab. inż. Krzysztof Boryczko, prof. nadzw. AGH

Kraków 2014

OŚWIADCZENIE AUTORA PRACY

OŚWIADCZAM, ŚWIADOMY(-A) ODPOWIEDZIALNOŚCI KARNEJ ZA POŚWIADCZENIE NIEPRAWDY, ŻE NINIEJSZY PROJEKT WYKONAŁEM(-AM) OSOBIŚCIE I SAMODZIELNIE W ZAKRESIE OPISANYM W DALSZEJ CZĘŚCI DOKUMENTU I ŻE NIE KORZYSTAŁEM(-AM) ZE ŹRÓDEŁ INNYCH NIŻ WYMIENIONE W DALSZEJ CZĘŚCI DOKUMENTU.

. . . . . .

PODPIS

Spis treści

1. Cel prac i wizja produktu

Symulacja czasu rzeczywistego zjawisk fizycznych jest zagadnieniem złożonym zarówno od strony teoretycznej jak i od strony implementacyjnej. Aby uzyskać realistycznie wyglądający efekt wymagane jest opracowanie szczegółowego modelu fizycznego danego zjawiska, zapisanie go w taki sposób aby umożliwić jego implementację i na końcu olbrzymia moc obliczeniowa aby móc podziwiać efekty działania modelu w czasie rzeczywistym. Pierwsze dwa kroki są zależne od nas i możemy je do woli szlifować, jednak w obliczu zbyt małej mocy procesorów jedyny możliwy efekt końcowy do zbiór danych wyjściowych otrzymywany po wykonaniu wszystkich obliczeń - rzecz bezużyteczna w symulacjach czasu rzeczywistego takich jak na przykład gry komputerowe.

W symulacji zjawisk fizycznych często występuje konieczność wykonywania dużej ilość prostych, powtarzalnych obliczeń dla różnych elementów układu. Przykładem może być symulacja zachowania źdźbeł trawy w szerokim polu. Mogą być ich miliony, a dla każdego z nich musimy wykonać podobne obliczenia zależne od takich czynników jak kierunek wiatru, nacisk otaczających elementów czy ukształtowanie podłoża. CPU nie będzie w stanie nadążyć przy tak olbrzymiej liczbie elementów nawet jeśli odpowienio wykorzystamy wszystkie dostępne rdzenie.

General purpose computation on graphics processing unit (GPGPU) to nazwa techniki wykonywania obliczeń na procesorach graficznych, która zyskuje na popularnośći w ostatnich latach. Wynika to z tego, że w nowoczesnych GPU wykorzystywany jest model obliczeniowy SIMT (single instruction, multiple threads), który w połączeniu z ogromną w stosunku do CPU ilością rdzeni (setki, czy nawet tysiące) sprawia, że procesory graficzne mogą wykonać duże ilości obliczeń równolegle.

Zjawisko fizyczne, które będzie nas interesować w naszej pracy to zachowanie cieczy znajdującej się za tamą w zamkniętym zbiorniku w przypadku przerwania tejże tamy i wylewania się cieczy do zbiornika. Istnieje wiele metod symulowania cieczy - dissipative particle dynamics (DPD), smoothed particle hydrodynamics (SPH) czy smoothed dissipative particle dynamics (SDPD), każda mająca swoje zalety i wady.

Zadaniem naszej pracy jest implementacja metody SPH w technice GPGPU oraz stworzenie aplikacji, która umożliwi wizualizację zachowania cieczy w trakcie wykonywania symulacji przerwania tamy. Wiąże się ona z koniecznością opanowania technologii OpenCL umożliwiającej GPGPU oraz OpenGL umożliwiającej renderowanie uzyskanych przez nas wyników.

2. Zakres funkcjonalności

Nasz projekt to aplikacja wykorzystująca przede wszystkim technologie OpenCL i OpenGL, w której można wyróżnić front-end (tj. wizualizację wykonaną przy pomocy OpenGL) oraz back-end (obliczenia, które symulują zachowanie cieczy). Jako, że obie wymienione technologie wykorzystują procesory graficzne, komunikacja między back-endem a front-endem została zrealizowana poprzez współdzielone bufory na karcie graficznej.

Aplikacja umożliwia poruszanie się po wirtualnym świecie przy pomocy klawiszy WSAD oraz lewego przycisku myszy, restartowanie symulacji przy pomocy prawego przycisku myszy oraz dobieranie parametrów symulacji takich jak lepkość czy gęstość cieczy etc.

3. Wybrane aspekty realizacji

- Do stworzenia okna i obsługi interakcji użytkownika z aplikacją posłużyła nam biblioteka GLFW.
- Skonstruowany został prosty framework, który umożliwił nam przyjemne mapowanie interakcji użytkownika na zachowania wewnątrz aplikacji. Pomogła nam w tym biblioteka Boost a w szczególności Boost.Signals2 umożliwiająca implementacją systemu zdarzeń.
- Do renderowania wykorzystaliśmy bibliotekę OpenGL. Kluczowa w osiągnięciu dobrych wyników okazał się dobór odpowiedniej techniki rysowania sfer zwanej “impostor sphere”.
- Do GPGPU wykorzystaliśmy bibliotekę OpenCL. Zdecydowaliśmy się korzystać z API w języku C a nie C++ ze względu na lepszą dokumentację.
- Konieczna była komunikacja między OpenGL i OpenCL. Wykorzystaliśmy do tego część OpenCL zwaną OpenCL/OpenGL interop.
- Całość projektu była zarządzana przy pomocy CMake. Wykorzystaliśmy tylko wieloplatformowe otwarte bibliotki co sprawia, że nasza aplikacja działa zarówno na systemach Linuks jak i Windows.

4. Organizacja pracy

Pierwsze rozmowy z klientem składały się głównie z omawiania tematu pracy, tłumaczenia modeli fizycznych oraz omawiania sposobu implementacji.

W związku z tym, że GPGPU było dla nas czymś nowym pierwsze tygodnie poświęciliśmy na naukę technologii OpenCL oraz na doszlifowanie wiedzy o OpenGL. Dalsze prace podzieliliśmy na trzy iteracje.

Pierwszą iteracją było stworzenie aplikacji okienkowej, która umożliwiała wyświetlanie przebiegu symulacji oraz stworzenie atrapy symulacji w OpenCL, ponieważ bez tego nie mieliśmy możliwości weryfikowania uzyskanych wyników.

W drugiej iteracji zaimplementowaliśmy algorytm SPH korzystając z OpenCL. Wynikiem drugiej iteracji była prawie w całości działająca aplikacja, niemniej jednak były pewne błędy, zarówno implementacyjne jak i wynikające z przyjętego modelu, które musiały zostać wyeliminowane.

Trzecia iteracja to poprawki w modelu obliczeń, debuggowanie kodu oraz dodatki umożliwiające łatwiejszą interakcją użytkownika z aplikacją oraz redagowanie dokumentacji.

Wizualizacji została napisana przez Dawid Romanowskiego natomiast symulacja została w większości napisana przez Wojciecha Czarnego. Ze względu na poziom skomplikowania

algorytmu SPH w trakcie implementacji symulacji korzystaliśmy z technik pair programming i rubber duck debugging.

W ostatniej iteracji Wojciech Czarny zajmował się naprawianiem usterek i testowaniem natomiast Dawid Romanowski redagowaniem dokumentacji ze zebranych w trakcie prac notatek.

5. Wyniki projektu

W wyniku pracy nad projektem powstały:

- aplikacja do symulowania zachowania cieczy metodą SPH
- dokumentacja składająca się z dokumentacji procesowej, technicznej i użytkownika.

Materiały źródłowe