Krzysztof Misan

Podsumowanie pracy nad projektem IFS generator

Na czym polegał projekt:

Projekt polegał na stworzeniu generatora fraktali IFS na podstawie algorytmu flam3 opisanego w publikacji z 2003r. https://flam3.com/flame_draves.pdf

Celem było:

-zaimplementowanie algorytmu w języku Python przy użyciu CPU

-zaimplementowanie algorytmu w języku Python przy użyciu obliczeń ogólnego zastosowania CUDA, wykonywanych przy użyciu GPU.

-stworzenie przejrzystego interfejsu, który pozwalałby na operowanie zmiennymi algorytmu

Użyte technologie, pakiety, biblioteki:

-używanym językiem był Python w wersji 3.7

-biblioteką graficzną był SFML, który stanowił warstwę abstrakcji nad OpenGL

-logiczną biblioteką graficzną był nuklear

-do obliczeń na gpu wykorzystano pyCuda

-do bindingów Python-C wykorzystano cffi oraz Cython

Stopień realizacji:

Wykonano wszystkie założenia przedstawione w początkowym konspekcie.

Napotkane problemy:

Największym problemem napotkanym podczas realizacji projektu było rozwiązanie konfliktów występujących między bibliotekami oraz stworzenie struktury programu umożliwiającej ich wzajemną współpracę. W programie przyjęto niekonwencjonalną myśl przewodnią, która zakładała, że żadna z bibliotek poza SFML nie może odwoływać się do pipeline'u OpenGL lub tworzyć nowej jego instancji. Spowodowało to szereg trudnych do rozwiązania utrudnień. Przykładowo biblioteka nuklear musiała zostać przepisana do Pythona w wersji 3. Dodatkowo, ze względu na pochodzenie bibliotek SFML, CUDA oraz nuklear pojawiła się potrzeba bliskiej współpracy języka C oraz Pythona, która została rozwiązana za pomocą cffi i Cythona.

Mocne strony projektu:

Uważam, że mocnymi stronami projektu są przede wszystkim: czytelny, responsywny interfejs oraz szybkie operacje na GPU.

Słabe strony projektu:

Myślę, że słabą stroną projektu jest przede wszystkim czytelność kodu. Ze względu na bardzo dużą ilość bindingów do kodu C oraz stosowanie silnych typów tego języka. Miejscami zaburza to składnię Pythona i następuje wymieszanie obu języków. Tak więc ścisła współpraca Pythona z C jest zarówno plusem (prędkość), jak i minusem (czytelność kodu).

Co można było zrobić inaczej, lepiej:

Przede wszystkim można było lepiej odseparować oba używane języki poprzez zapewnienie dodatkowej warstwy abstrakcji przez Pythona. Ponadto nie jestem zadowolony z prędkości działania kodu w Pythonie, który operuje na CPU, operacje kopiowania danych między RAM i GPU to bottleneck całego algorytmu, należałoby stworzyć dodatkowy kod przy użyciu Cythona, który zwiększyłby efektywność działania. Na zakończenie, myślę, że nieco inaczej podszedłbym do kwestii odseparowania OpenGL, użycie ui w wersji logic only to niewątpliwy atut, jeżeli chodzi o enkapsulację ale wymusza ogromną ilość dodatkowego kodu, który niekoniecznie ma sens dla mniejszych projektów.

Co można dodać do projektu:

Gdybym miał dalej rozwijać projekt, przede wszystkim skupiłbym się na zużyciu pamięci przez algorytm, chciałbym umożliwić podział obliczeń i zastosowanie kolejki obliczeniowej. Drugą rzeczą, której niewątpliwie brakuje, to rozwinięcie wartości wynikowych do 32 bitów. Myślę, że ciekawą funkcjonalnością byłaby również możliwość budowy poklatkowej animacji zastosowanych transformacji.

Ocena projektu:

Uważam, że końcowy efekt jest zadowalający. Wszystkie początkowe założenia zostały zrealizowane w określonym czasie, a projekt ma na tyle solidną podstawę, że jego dalszy rozwój nie byłby dla mnie problemem.