Concurrency Report

Robin Sikkens 4228189 & Mathijs Bangma 4135520

Om het map pattern toe te passen hebben wij tiling toegepast - we vinden de aspect ratio van het scherm door de grootste gemene deler van de screen.width en screen.height te vinden. Vervolgens deelden we de schermwaarden daardoor om kleine tiles te krijgen. Vervolgens konden we een Parallel.For loop over die tiles draaien, waarin elke pixel van die tile berekend werd en de X,Y waarde gecorrigeerd naar de plaats van die tile. Zo heb je al redelijk wat data locality.

We probeerden ook een morton-curve over die tiles toe te passen (de tile*tile was vierkant, dus dat zou niet al te lastig moeten zijn), maar dat real-time berekenen bleek te langzaam te zijn. Van tevoren een lookup-tabel genereren was een optie, maar aangezien het ons zo lang duurde om OpenCL werkend te krijgen was dat geen prioriteit geweest, en dat zit er ook uiteindelijk niet in.

De random number generator klaar maken voor parallellisatie bleek ook vrij eenvoudig - een simpele [ThreadStatic] ervan maken bleek alle problemen al op te lossen.

We dachten ook een vrij lange tijd dat de versnelling lang niet goed genoeg was (op onze intel i5 4690K 3.50 Ghz kregen we ~320% van de verwachte 350% van TomJudge), maar toen we op de vakpagina zagen dat dat een probleem van Intel-processoren kon zijn vroegen we iemand die wél genoeg versnelling had of we onze code even op zijn computer mochten draaien, en toen bleek ons programma 427% van de verwachte 405% versnelling te hebben - dat was een hele opluchting.

We hebben de GPU code getest op een systeem met een intel i5 4690K 3.50 Ghz processor, en een GTX 970 grafische kaart. In vergelijking tot de referentie hebben we een performance versnelling van 1378%. Omdat we niet heel erg veel tijd hadden nadat we openCL werkende hadden gekregen is het bijna het gehele programma uitgerekend in de kernel. De kernel genereert zelf een ray en tracet die tot een diepte van maximaal 20 waarna die resulterende waarde bij de accumulator wordt gevoegd en wordt omgezet naar een IntegerRGB voordat hij wordt geschreven naar het scherm. De CPU wordt hier jammergenoeg alleen gebruikt om *scale* uit te rekenen en naar de GPU te sturen en om de schermbuffer uit de GPU op te halen. Als we meer tijd hadden gehad om ons systeem heterogener te maken hadden we het genereren van de rays parallel op de CPU uitgevoerd.