NoisyCuda

Introduzione

NoisyCuda propone di fornire una libreria ad alte prestazioni per la generazione di Perlin Noise, sfruttando l'elaborazione parallela offerta dalle GPU tramite CUDA e ottimizzazioni a livello di CPU tramite istruzioni SIMD. L'obiettivo principale è accelerare significativamente la produzione di mappe di rumore utilizzate in grafica procedurale, simulazioni fisiche e applicazioni di rendering, mantenendo alta la qualità visiva e la coerenza del rumore.

La metodologia prevede la parallelizzazione delle operazioni in CUDA, mentre le istruzioni SIMD vengono impiegate per ottimizzare i calcoli vettoriali e ridurre i colli di bottiglia della CPU.

Il progetto include benchmark comparativi tra implementazioni tradizionali su CPU e la versione accelerata su GPU, evidenziando guadagni in termini di throughput e scalabilità.

L'implementazione risultante fornirà uno strumento efficiente e modulare, facilmente integrabile in pipeline grafiche real-time e software di simulazione.

Struttura del progetto

Il progetto è stato strutturato in modo altamente modulare, dando la possibilità all'utente di scegliere durante la compilazione la tecnologia (pura cpu, cpu SIMD, CUDA) e la relativa versione da utilizzare.

noisyCuda/include/

Il folder /include/ contiene tutti gli header principali utilizzati nel progetto. Vengono utilizzati principalmente per 2 scopi:

- rendere uniforme la firma della generazione di un Noise

- dichiarare delle strutture dati globali (come il Vector2D), necessarie per rappresentare nel modo migliore dati come i vettori unità dei chunk di generazione.

noisyCuda/src/

Il folder /src/ viene diviso nelle 3 tecnologie implementate.

Ogni sottocartella presenta la lista delle varie implementazioni sviluppate, in ordine di versione. La prima versione offre semplicemente un programma dove sono state implementate tutte le funzionalità previste deall'applicativo.

Le versioni successive tendono a sfruttare i vantaggi offerti dalla tecnologia specifica per ottimizzare throughput e tempo totale d'esecuzione.

Algoritmo standard di generazione Perlin Noise

Abbiamo utilizzato questa serie di passaggi standard per poter ottenere un algoritmo funzionante su cui basarci per le nostre accelerazioni, offrendo anche un modo per studiare le varie particolarità teoriche e pratiche dell'algoritmo fin dalla sua versione più diffusa.

Di seguito si illustrano i vari step che permettono di ottenere un Perlin Noise:

1. Definizione della griglia di punti di controllo

Si crea una griglia regolare di punti nello spazio in cui il rumore sarà generato. Ogni punto della griglia sarà associato a un vettore di gradiente casuale, normalmente di lunghezza unitaria, che determinerà l'influenza locale sul valore del rumore.

2. Calcolo delle coordinate relative

Per ogni punto nello spazio in cui si vuole calcolare il rumore, si determina la posizione relativa rispetto ai vertici della cella della griglia in cui cade. Questa posizione viene espressa come coordinate frazionarie tra 0 e 1, che serviranno per interpolare i valori tra i vertici della cella.

3. Calcolo del prodotto scalare con i gradienti

Per ciascun vertice della cella, si calcola il vettore che va dal vertice al punto di interesse e si effettua il prodotto scalare con il vettore gradiente assegnato al vertice stesso. Questi prodotti scalari rappresentano l'influenza dei vertici sul punto e costituiscono i valori base da interpolare.

4. Applicazione della funzione di fade

Si applica una funzione di fade (tipicamente 6t^5 - 15t^4 + 10t^3) alle coordinate frazionarie. Questa funzione serve a rendere più morbidi i passaggi tra le celle, evitando discontinuità o angoli troppo netti nel rumore risultante.

5. Interpolazione dei valori

Si interpolano i prodotti scalari calcolati nei vertici usando le coordinate fade. Nel Perlin Noise classico si usa l'interpolazione lineare o la funzione di interpolazione "lerp" in più dimensioni, combinando prima le interpolazioni lungo un asse e poi lungo l'altro (nel caso 2D o 3D).

6. Combinazione dei risultati

Dopo l'interpolazione, si ottiene un singolo valore di rumore per il punto considerato. Questo valore può essere normalizzato per rientrare in un intervallo standard, ad esempio [-1, 1], e può essere combinato con altre frequenze per generare Perlin Noise "octave" più complesso.

7. Ripetizione per l'intera area

Si ripete il calcolo per tutti i punti dello spazio in cui vogliamo generare il rumore. Questo produce una mappa di valori continui e coerenti, che rappresenta il Perlin Noise classico pronto per essere usato in grafica procedurale, simulazioni fisiche o altri contesti.

Strumenti Aggiuntivi

La repository di NoisyCuda è fornita di alcuni strumenti che arricchiscono l'utilizzo del programma di generazione. In particolare, nella directory /tests/ possiamo trovare:

NoisyStudio

Un piccolo frontend grafico che permette di evitare l'uso del programma tramite linea di comando, e offrendo un modo rapido e user-friendly per controllare al meglio la generazione di nuovi noise.

StartCudaProfiling.sh

Un codice shell che permette di eseguire la generazione di una texture e di far analizzare i dati registrati al profiler ufficiale di Nvidia. Strumento ottimo per la raccolta efficace di informazioni sull'esecuzione del programma.