

Progetto di Computer Graphics : libreria di Noise

Andras Boguslaw Arancio

January 2023

1 Introduzione

Il progetto che ho scelto è stata la libreria di Noise procedurali realizzati in C++. Ho iniziato creando un programma in grado di rappresentare immagini 2D dei noise; ho poi applicato i noise come texture di sfere 3D mediante trilinear interpolation, a tal fine ho utilizzato il raytracer di Peter Shirley *In One Weekend*¹, per via della sua facilità d'utilizzo e della dimestichezza che ho acquisito con questo raytracer. Una collezione di immagini da me realizzate, ottenute mediante variazioni frattali dei Noises, è disponibile nella cartella Risultati.

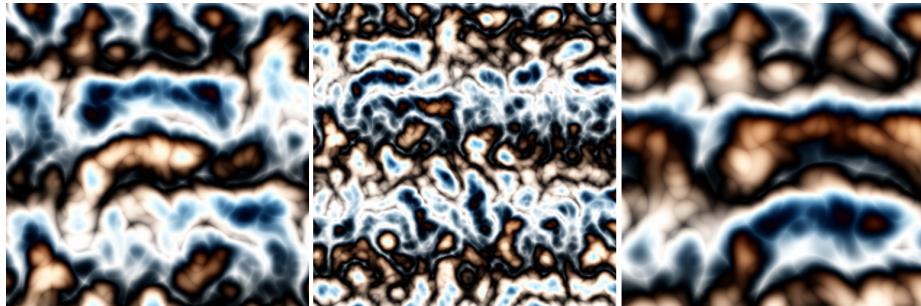


Figure 1: Frequenze diverse di una variante che ho realizzato

2 Value noise e Perlin noise

I primi passi che ho mosso sono stati diretti verso la realizzazione del più basile dei noise: il Value noise. In questa tipologia di noise, i valori di ciascun pixel vengono rimappatti su una griglia in cui ad ogni posizione di un numero intero è associato un valore randomico; il valore del punto ricercato sarà uguale all'interpolazione lineare fra gli interi a lui più vicini, utilizzando la distanza

¹P. Shirley, *Ray Tracing In One Weekend Series* : <https://raytracing.github.io/>

che separa il punto ricercato dagli interi come fattore interpolante. La scelta del fattore interpolante è fondamentale per la resa finale del noise, per questo, come illustrato dalle diverse guide che ho seguito per la realizzazione di Value noise e Perlin noise², è possibile applicare delle funzioni di remapping, al fine di ammorbidire l'interpolazione. Il Perlin noise è concettualmente analogo al Value noise, a differenza di questo ha dei vettori anziché dei valori randomici nelle posizioni degli interi. Gli estremi dell'interpolazione sono rappresentati dal dot product fra il vettore randomico associato a ciascun intero più vicino e il vettore compreso tra il punto di cui si desidera calcolare il valore e il rispettivo intero. Il risultato è un noise generalmente più morbido e organico. L'interpolazione trilineare utilizza una griglia tridimensionale, il passaggio da 2D a 3D è stato facile da attuare.

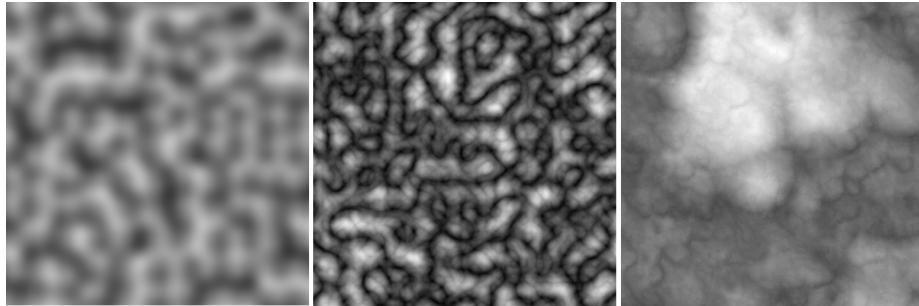


Figure 2: Basic Perlin Noise, Turbulence ed una mia variante

3 Voronoi

Il concetto dietro il Voronoi è il seguente. Ciascun punto di cui si vuole calcolare il valore viene rimappato su una griglia; il centro di ciascuna cella che compone la griglia viene scostato di un certo displacement. Il punto in input verrà associato al centro scostato a lui più vicino. In tal modo si ottiene un pattern organico molto diffuso in natura. Le variazioni del Voronoi che ho implementato sono lo Smooth Voronoi, il Voronoi Distances e il Voronoise, un mix fra Voronoi e Noise. Come guide implementative ho seguito gli articoli di Inigo Quilez³. Nel caso dello smooth voronoi, il valore in output della funzione è, anziché la distanza dal centro scostato più vicino come nel Voronoi standard, una media pesata delle distanze dai 9 centri scostati più vicini, questa modifica rende i bordi delle celle sfocati. Nel Voronoi distances(o edges), si cercano dapprima le due celle più vicine al punto in input; una volta individuati questi

²Le guide di Scratchapixel, in particolare *Procedural Generation of Virtual Worlds* : <https://scratchapixel.com/> e Building up Perlin noise di Andrew Kensler: <http://eastfarthing.com/blog/2015-04-21-noise/>

³Inigo Quilez::articles, in particolare Procedural Noises : <https://iquilezles.org/articles/>

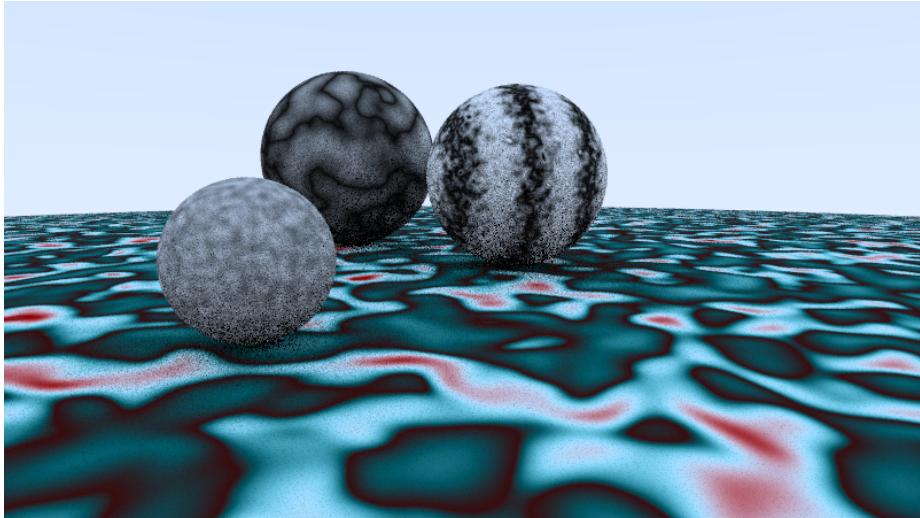


Figure 3: Turbulence, Marble e mia variante con TLI.

due centri, computare la distanza dal border più vicino è solo questione di algebra dei vettori. Dal momento che sia il Noise che il Voronoi, come abbiamo esplicato nelle descrizioni precedenti, sono basati su delle griglie, si potrebbe tentare di generalizzare i due pattern procedurali in un unico più generico⁴. Il Voronoise deve disporre di due parametri fondamentali: uno che regoli il displacement dei centri, l'altro che regoli il disurbo; a tal fine è possibile sfruttare le proprietà dell'implementazione dello smooth Voronoi, facendo una media aritmetica piuttosto che pesata delle distanze dai centri discostati, il che ci da una interpolazione lineare delle distanze.

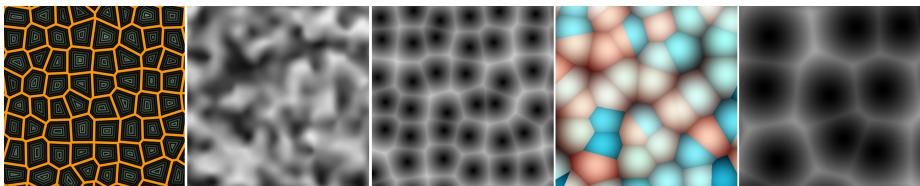


Figure 4: Voronoi distances, Voronoise, Voronoi, Voronoi Colored, Smooth Voronoi

⁴Inigo Quilez, Voronoise Intro : <https://iquilezles.org/articles/voronoise/>

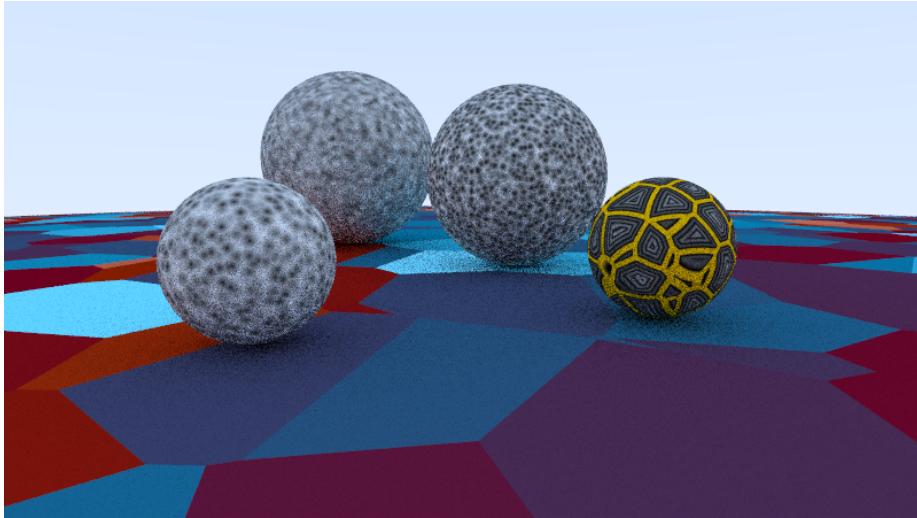


Figure 5: Voronoi standard, Smooth, Edges, Colored e Voronoise con TLI

4 Fractal sums

Per ottenere effetti più gradevoli e realistici, è possibile effettuare delle somme frattali di uno o più tipologie di noise⁵. Questo consiste nel sommare fra di loro noises con frequenze e ampiezze diverse, ciascun componente di questa somma verrà chiamato layer o ottava(quest'ultimo termine non è però corretto in generale). In linea di principio ciò che si vuole ottenere è una distribuzione frattale del noise, in cui vi è una ricorrenza, nei dettagli del noise, del noise stesso.

Le funzioni di hashing che ho utilizzato nei vari Noise laddove erano necessarie sono tutte funzioni "note", alla maggior parte delle quali sono giunto studiando il codice di Inigo Quilez. Le eccezioni sono opportunamente citate mediante commenti nei file sorgente. I risultati che si possono ottenere distorcendo e sommando fra loro i layer sono molto interessanti, mostro alcuni esempi che ho realizzato.

5 I programmi

Come anticipato nell'Introduzione, ho realizzato le immagini 2D e 3D in due programmi separati. Per testare il programma 2D, scommentare nel main il

⁵Gli articoli che ho studiato sono : <https://iquilezles.org/articles/fbm/> e <https://scratchapixel.com/lessons/procedural-generation-virtual-worlds/procedural-patterns-noise-part-1/simple-pattern-examples.html>

⁷Warped Clouds by clayjohn : <https://www.shadertoy.com/view/4tfXzl>

⁹Warped Texture Perlin di manu210404 : <https://www.shadertoy.com/view/ttsfzH>

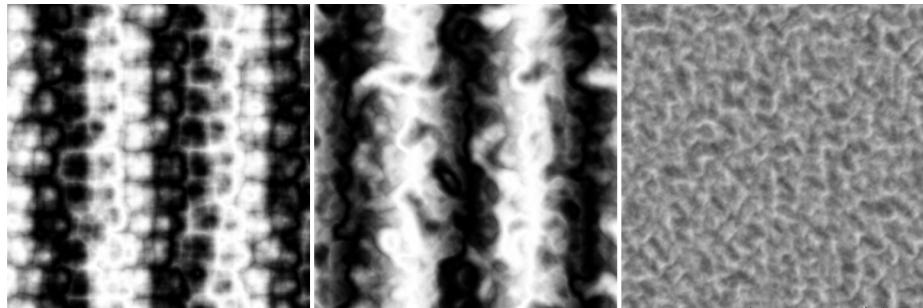


Figure 6: Pelle di serpente, Marmo e una variazione frattale di Perlin Noise.

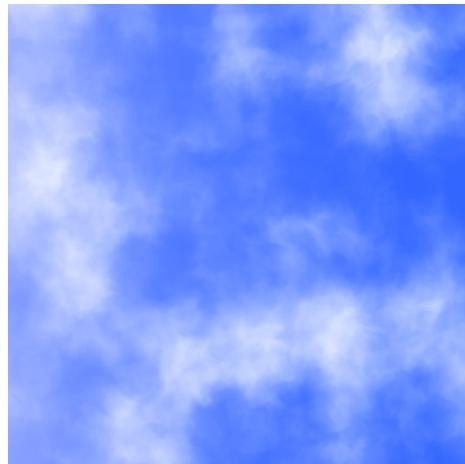


Figure 7: Nuvole in cielo. Per la realizzazione ho preso spunto dall'utente di ShaderToy clayjohn⁷.

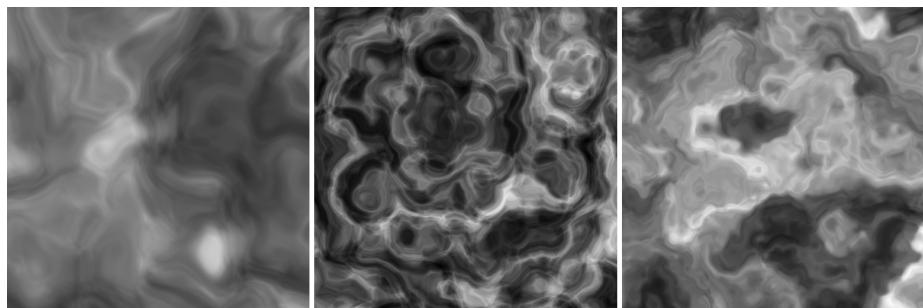


Figure 8: Effetto fluido ottenuto distorcendo e applicando variazioni frattali al Perlin Noise. Per la realizzazione ho preso spunto dall'utente di Shadertoy manu210404⁹. Al centro e a destra due variazioni in cui ho aggiunto del turbulence alla somma, utilizzato il Voronoise anziché il Perlin, e apportato ulteriori modifiche agli uv.

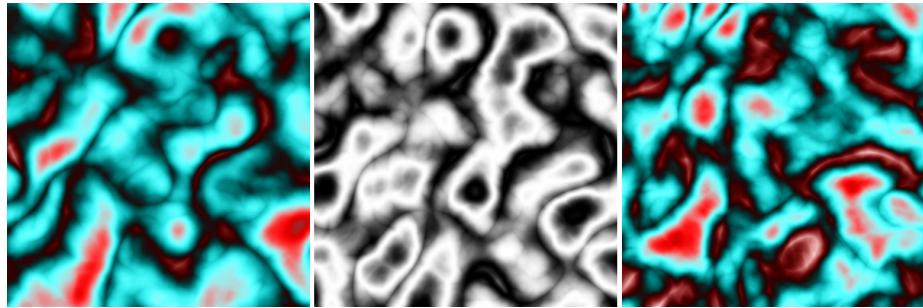


Figure 9: Altri risultati che ho ottenuto.

codice del Noise che si desidera ottenere, compilare il programma con il proprio compilatore e scrivere l'output del programma in un file.ppm :

```
./main > img.ppm
```

Per la realizzazione dei Noise applicati ad oggetti 3D con trilinear mapping ho utilizzato il RayTracer InOneWeekend. Eseguendo il programma allo stesso modo della versione 2D, si otterrà un'immagine simile a quelle presenti in questo documento. Nel caso si voglia cambiare texture, è possibile farlo ed ho aggiunto nel main dei commenti che illustrano come farlo.