

Laurea triennale in Ingegneria Gestionale classe L8

A.A. 2023/2024

Sessione di Laurea Luglio 2024

**Generazione procedurale di heightmaps tramite interpolazione di rumori**

Relatore: Averta Giuseppe Bruno

Candidato: s213405 Alessio Dami

# **1 - Proposta di progetto**

# Studente proponente

s213405 Dami Alessio

# Titolo della proposta

Algoritmo di generazione procedurale di heightmap 2D tramite interpolazione del rumore di Perlin e diagramma di Voronoi

# Descrizione del problema proposto

Il rumore di Perlin è un tipo di gradient noise (rumore creato tramite funzioni gradiente) che è utilizzato nella creazione di texture, mappe, superfici e molto altro. Ha i vantaggi di essere pseudo-casuale, graduale tra i suoi valori (senza sbalzi) e controllabile tramite i suoi attributi. L'algoritmo però è omogeneo, senza zone distinte tra loro.  
L'integrazione di un diagramma di Voronoi può risolvere questo problema, interpolando i valori normalmente ottenuti dal rumore con quelli dati da una media pesata delle distanze di ogni singola coordinata rispetto ai suoi nodi più vicini

# Descrizione della rilevanza gestionale del problema

I classici algoritmi di generazione rumore sono utili al' ingegnere gestionale quando si lavora su mappe che possono rappresentare il territorio.

Le varie celle del diagramma di Voronoi possono essere assimilabili ai vari distretti, zone o quartieri di una città, o similmente varie zone di una provincia e così via. Queste zone rappresentano la presenza più o meno elevata di una determinata caratteristica, come ad esempio la copertura di rete, o la presenza di centri di distribuzione e logistica.  
A queste zone può essere applicato quindi un rumore di sottofondo per simulare una casualità all'interno della zona.  
L'algoritmo quindi potrebbe essere utile nella simulazione a priori, e rimane comunque utile nel suo scopo originale di generazione texture, mappe, shader e simili utilizzi.

# Descrizione dei data-set per la valutazione

L'algoritmo utilizza un singolo database formato dall'insieme dei diagrammi di Voronoi sotto forma di insieme dei seed che li generano.  
Il database è stato popolato casualmente, ed il programma permette di recuperare uno specifico diagramma tramite ricerca dei nodi.

# Descrizione preliminare degli algoritmi coinvolti

Il primo passo è la generazione del diagramma di Voronoi dati un insieme di punti (siano essi presi dal database o inseriti manualmente come nuovi).  
Dopodiché il programma, per ogni coordinata del piano, trova i K nodi/semi più vicini (numero modificabile) e applica una media pesata delle distanze della coordinata da questi K nodi (più vicino è il nodo più peso avrà nella media). Questo passaggio permette, maggiore siano i nodi presi in considerazione, di trascurare via via i bordi tra le varie celle del diagramma che sono tipiche di Voronoi.  
Salvate queste distanze pesate (e normalizzate nell'intervallo [0,1] per essere compatibili con altri rumori) viene generato, sullo stesso piano di uguali dimensioni, un rumore di Perlin secondo le sue caratteristiche (ampiezza, frequenza, ottave, etc.).  
Infine l'algoritmo attua un'interpolazione lineare tra i due piani (il piano delle distanze pesate normalizzate e il piano del rumore di Perlin) e fornisce il piano interpolato finale.

Durante questi passaggi c'è la possibilità di mostrare a schermo i risultati intermedi, come ad esempio l'heightmap data dalle distanze pesate normalizzate del diagramma di Vononoi, oppure il rumore di Perlin preso singolarmente, oppure il risultato interpolato finale, sia come scala di grigi (grayscale) sia come immagine colorata (ad esempio si assegnano dei colori blu, celeste, verde, marrone e bianco per simulare una mappa, con valori che vanno dal blu (0) che rappresenta il mare profondo al bianco (1) che rappresenta vette di montagna)

# Descrizione preliminare delle funzionalità previste per l’applicazione software

Sebbene non esaustive, queste sono le funzioni previste dall'applicativo:

Inserimento e reperimento di un diagramma di Voronoi dal database. Possibilità di mostrare a schermo il diagramma corrente. L'inserimento dei nodi avverrebbe tramite coordinate 2D

Modifica dei parametri usati nell'algoritmo da parte dell'utente per ottenere risultati in base alle proprie preferenze.  
Un elenco non esaustivo è:  
Numero di nodi da prendere in considerazione quando si fa la media pesata delle distanze,  
Clamping valori dell'heightmap delle distanze: possibilità di normalizzare valori oltre un certo intervallo (valori troppo scuri o chiari)  
Possibilità di invertire i valori delle distanze: un valore più vicino a un nodo sarà più scuro piuttosto che essere più chiaro.  
Attributi del rumore di Perlin: ampiezza, frequenza, ottave e persistenza

Inoltre, sarà data la possibilità di mostrare risultati (immagini) intermedi e salvarle.

# **2 – Descrizione dettagliata del problema affrontato**

La generazione procedurale è una tecnica fondamentale nella creazione di contenuti multimediali, utilizzata per generare texture, mappe e altri elementi. I principali vantaggi includono:

1. **Efficienza e Risparmio di Tempo**: La generazione automatica di contenuti riduce enormemente il tempo necessario per la creazione manuale di ogni dettaglio, permettendo agli sviluppatori di concentrarsi su altri aspetti del progetto.
2. **Varietà e Diversità**: La generazione procedurale consente la creazione di una vasta gamma di varianti dello stesso elemento, incrementando la diversità senza dover sviluppare ogni singola variante manualmente. Questo è particolarmente utile nei giochi, dove è necessario creare mondi vasti e dettagliati, e dove l’iterazione continua può richiedere molte varianti differenti.
3. **Scalabilità**: Con la generazione procedurale, è possibile creare contenuti che si adattano facilmente a diverse risoluzioni e dimensioni senza perdere qualità, rendendo questa tecnica ideale per applicazioni differenti ma simili.
4. **Ottimizzazione delle Risorse**: La generazione procedurale riduce la quantità di memoria necessaria a immagazzinare i dati, poiché gli algoritmi generano contenuti in tempo reale piuttosto che memorizzare i dati statici su disco.
5. **Innovazione e Creatività**: Gli algoritmi procedurali possono introdurre elementi imprevedibili e unici, portando a soluzioni innovative che potrebbero non emergere con metodi di creazione tradizionali.

Il più diffuso algoritmo per la generazione di questo tipo è il rumore di Perlin, ideato nel 1983 da Ken Perlin, un ingegnere grafico, per poi essere rifinito nel 2001.

Il rumore di Perlin è una funzione di rumore gradiente utilizzata nella generazione procedurale di texture e nella grafica computerizzata. Tra i suoi principali vantaggi troviamo la capacità di generare variazioni casuali lisce e continue, ideali per creare texture e terreni dall'aspetto naturale. È anche altamente personalizzabile, permettendo di regolare la frequenza e l'ampiezza per ottenere una vasta gamma di effetti dettagliati. Inoltre, il rumore di Perlin è efficiente dal punto di vista computazionale, rendendolo adatto per applicazioni in tempo reale come i videogiochi e graphic design.

Nonostante i numerosi vantaggi i rumori di Perlin, pur essendo lisci e continui, possono risultare troppo uniformi e privi di varietà, rendendoli meno adatti per rappresentare alcune forme di rumore naturale più irregolari e caotiche.

Per questo motivo ho deciso di sviluppare questa tesi, interpolando il rumore di Perlin con i diagrammi di Voronoi, suddivisioni del piano in regioni dove ogni regione contiene tutti i punti più vicini a un determinato nodo preesistente. Questa interpolazione permette di limitare gli svantaggi del rumore di Perlin, soprattutto la troppa uniformità.

**3 – Descrizione dei dataset utilizzati**

L’algoritmo utilizza un database formato da due dataset, l’insieme dei diagrammi di Voronoi e l’insieme dei nodi di tutti i diagrammi.

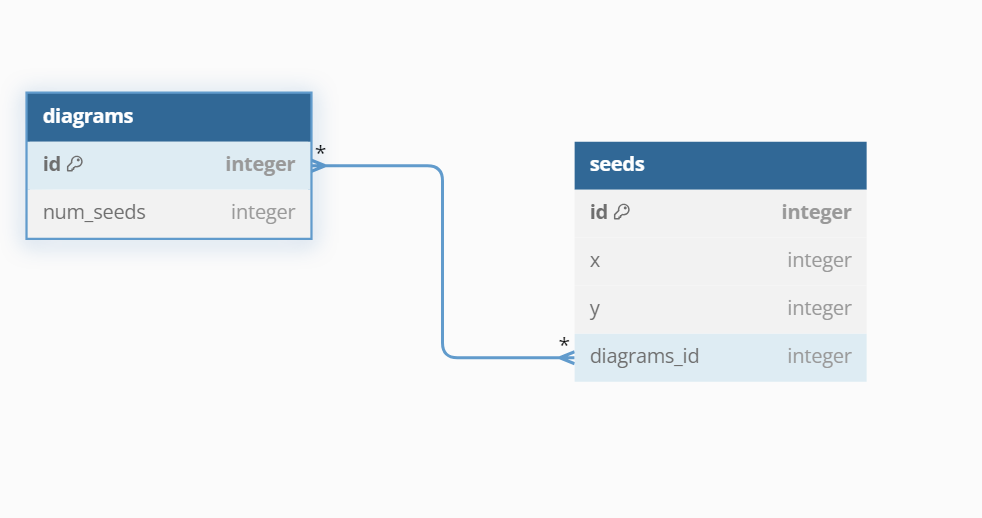
Il database è stato manualmente creato per l’uso in questo algoritmo. Sono stati creati 10 diagrammi di Voronoi per ogni numero di nodi possibile da 1 a 50, quindi 500 diagrammi totali e 12750 nodi totali. I nodi sono stati posizionati casualmente ipotizzando la dimensione dell’immagine di 1000x1000 pixels.

La tabella diagram contiene:

* id: intero e chiava primaria incrementale, rappresenta l’id del diagramma
* num\_seeds: intero, rappresenta il numero di nodi del diagramma

La tabella seeds contiene:

* id: intero e chiava primaria incrementale, rappresenta l’id del nodo
* diagram\_id: intero e chiave esterna che rappresenta l’id del diagramma a cui il nodo appartiene
* x: intero, rappresenta la coordinata x del nodo
* y: intero, rappresenta la coordinata y del nodo

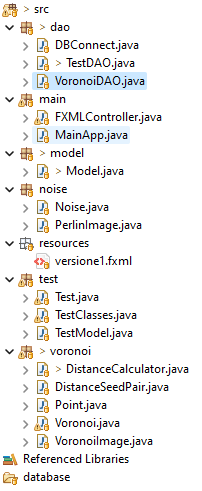


**4 – Descrizione delle strutture dati**

Il programma è stato realizzato con: la struttura MVC (Model-View-Controller) per quanto riguarda la parte applicativa, la struttura DAO (Data Access Object) per quanto riguarda la parte di interazione con database, e con il software Scene Builder per la realizzazione dell’interfaccia grafica.

Il progetto è costituito dai seguenti package principali:

* Package dao: contiene le classi per l’interazione con il database. DBConnect consente la connessione col database, VoronoiDAO integra quest’ultima classe per estrarre tutti i diagrammi di una certa dimensione sotto forma di lista di nodi.
* Package main: contiene la classe per far partire il programma, più la classe FXMLController che integra la classe Model e il file FXML per legare le funzioni all’interfaccia grafica
* Package model: contiene l’unica classe Model che contiene tutte le funzioni principali utilizzate dal programma in un unico posto.
* Package noise: contiene le classi relative all’algoritmo di Perlin, ovvero l’algoritmo in sé del rumore e quello che permette di creare un'immagine da questo.
* Package resources: contiene l’unico file FXML che descrive l’interfaccia grafica.
* Package voronoi: contiene le classi relative al diagramma di Voronoi, come il diagramma stesso, la classe Point che rappresenta i nodi, le classi per calcolare le distanze dei punti e la classe per rappresentare il diagramma sotto forma di immagine.

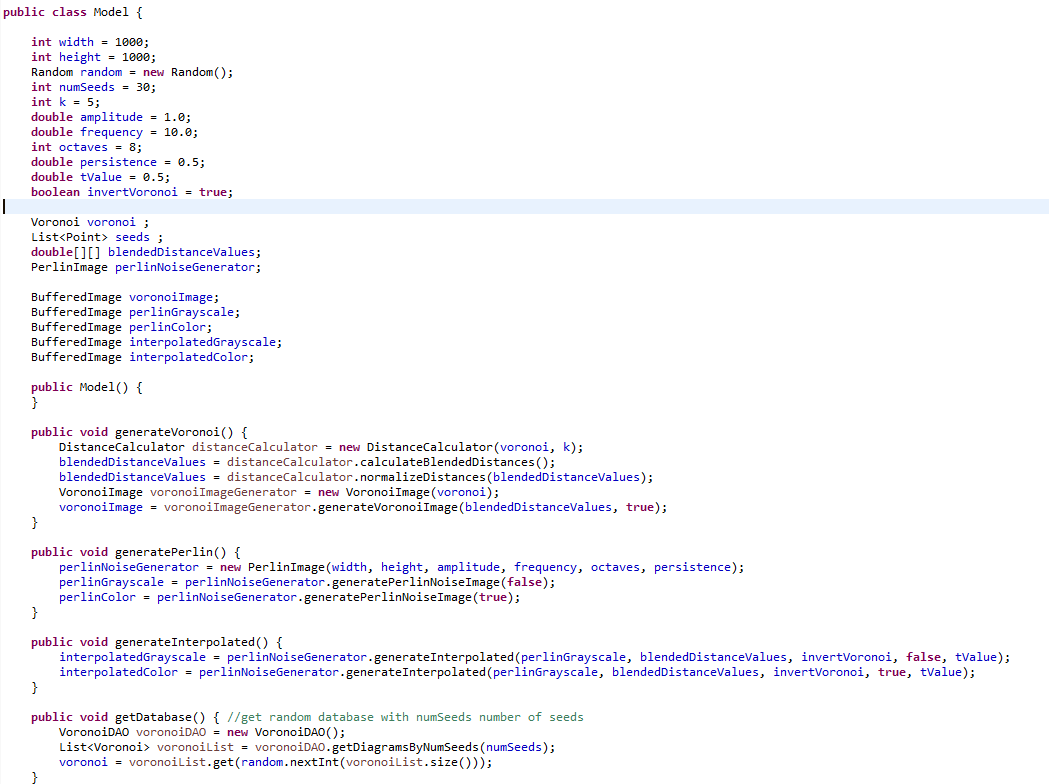


**5 – Diagramma e descrizione delle classi più importanti**

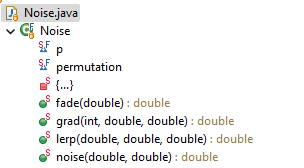
**Model:**



La classe Model contiene la logica applicativa che richiama tutte le classi. Permette di generare il diagramma di Voronoi con relativa immagine, generare il rumore di Perlin con relative due immagini (in scala di grigi o colorate), generare l’interpolazione tra questi due con relative immagini (scala di grigi o colorate) e infine richiamare il database per estrarre un singolo diagramma di Voronoi.

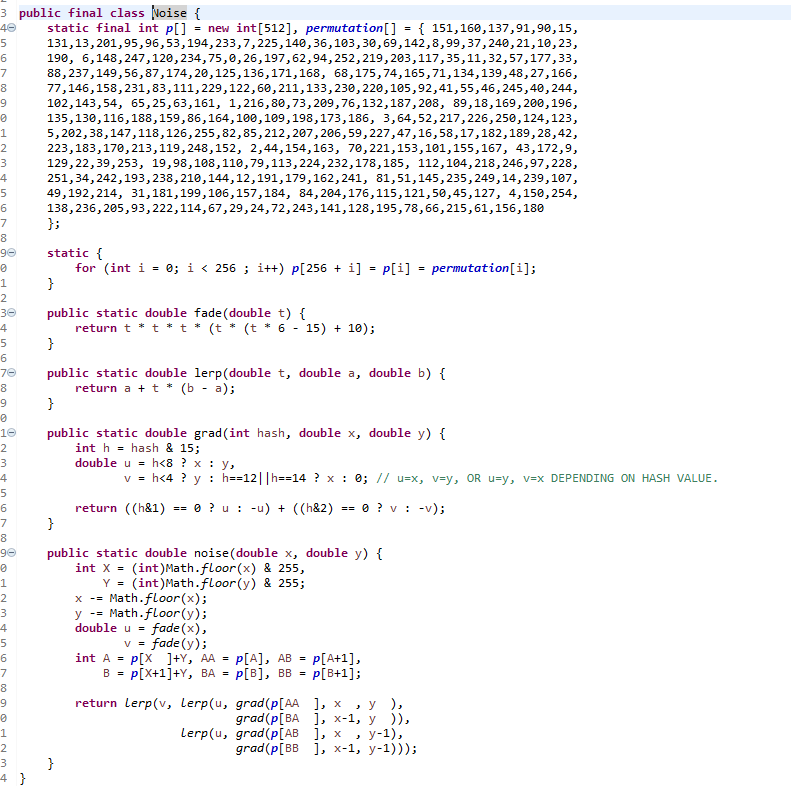


**Noise:**

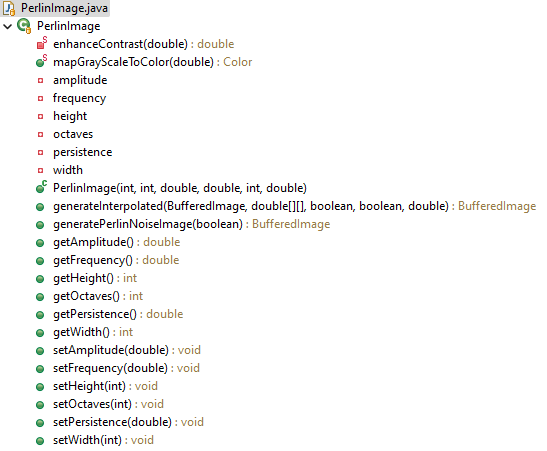


La classe Noise è presa dall’integrazione in Java fatta direttamente da Ken Perlin dell’algoritmo per la generazione del rumore.

Contiene una tabella di permutazioni che permette una pseudorandomità all’algoritmo. Contiene inoltre le funzioni fade (per fare lo smoothing), lerp (per interpolare linearmente tra 4 valori durante l’algoritmo), grad (che fornisce i 4 valori per semplificare) e la funzione noise stessa che applica l’algoritmo in sé.



**PerlinImage:**

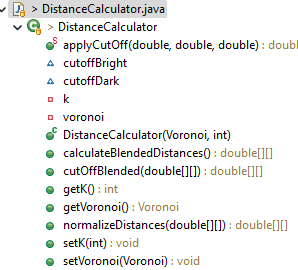


La classe PerlinImage contiene i dati per la modifica dell’algoritmo di Perlin (ampiezza, frequenza, ottave, persistenza) e contiene le due funzioni per la generazione delle immagini sia di Perlin (in scala di grigi e colorata) sia dell’interpolazione di Perlin e Voronoi (sempre in scala di grigi o colorate)

Queste funzioni infatti sono chiamate due volte, una con attributo boolean (coloured) FALSE, l’altro con TRUE.

Per la stampa a schermo questa classe utilizza la libreria java.awt.image e java.imageio .

**DistanceCalculator:**



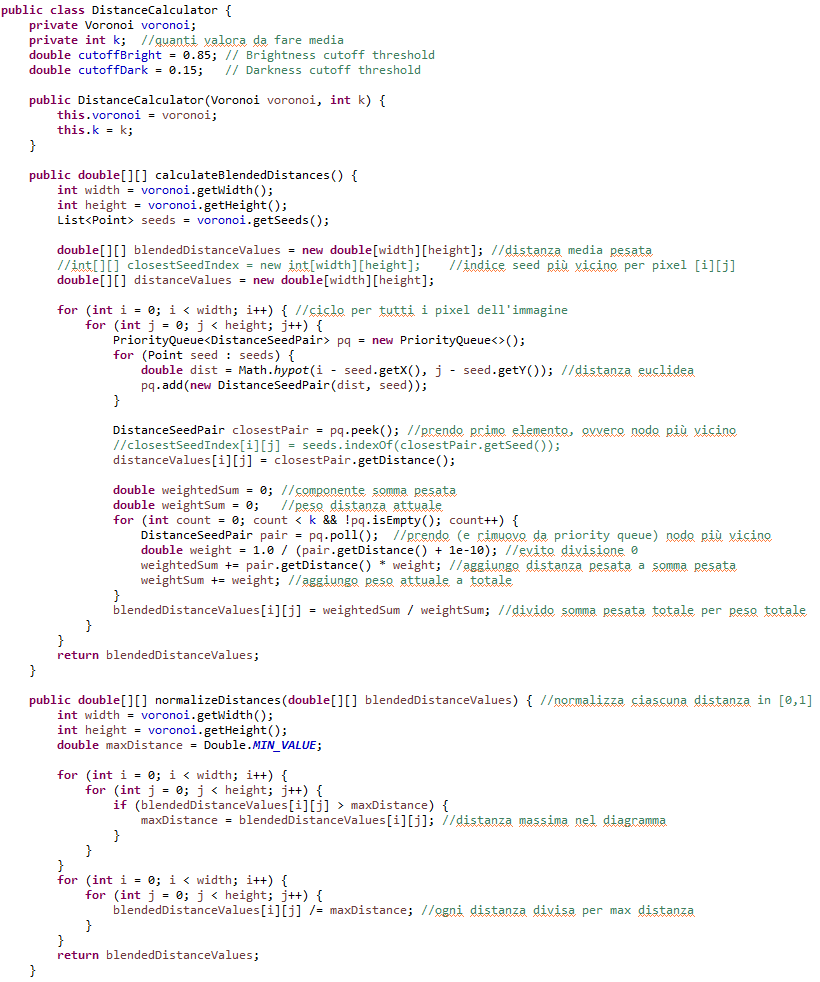
La classe DistanceCalculator, dato un diagramma di Voronoi, di calcolare le distanze di ogni pixel rispetto ai suoi K nodi più vicini (con K modificabile). Con valori di K maggiori di 3 i bordi tra le varie celle del diagramma di Voronoi iniziano a sparire, cosicché non possano influenzare il rumore di Perlin (dato che i bordi essendo tra poligoni sono rette uniformi).

La classe contiene 3 funzioni principali: blendedDistanceValues fornisce una matrice bidimensionale, pari al numero di pixel orizzontali per verticali.

Per ogni pixel calcola la media pesata delle distanze del pixel dai K nodi più vicini (pesata cosicché il nodo più vicino sia più influente di quello più lontano, con peso legato alla distanza di questi).

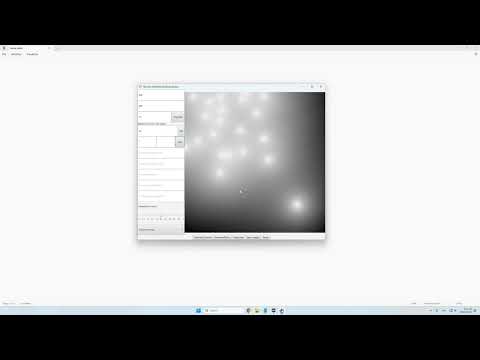
La classe normalizeDistance prende quindi questa matrice e normalizza i valori tra 0 e 1 compresi, affinché possano essere interpolati con i valori del rumore di Perlin, anch’essi di valore tra 0 e 1.

Infine, la funzione applyCutOff (e cutOffBlended) applicano un clamp (ovvero un limite massimo o minimo) affinché i valori non siano troppo chiari o scuri.

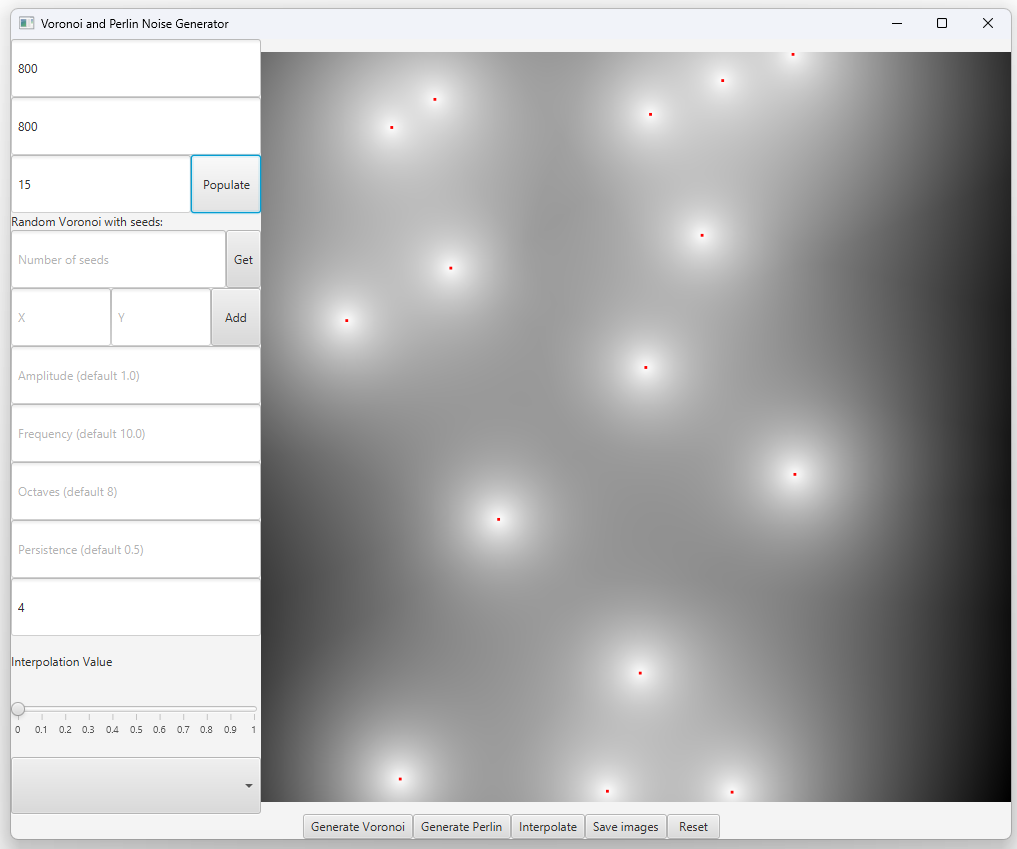


**6 – Interfaccia video**

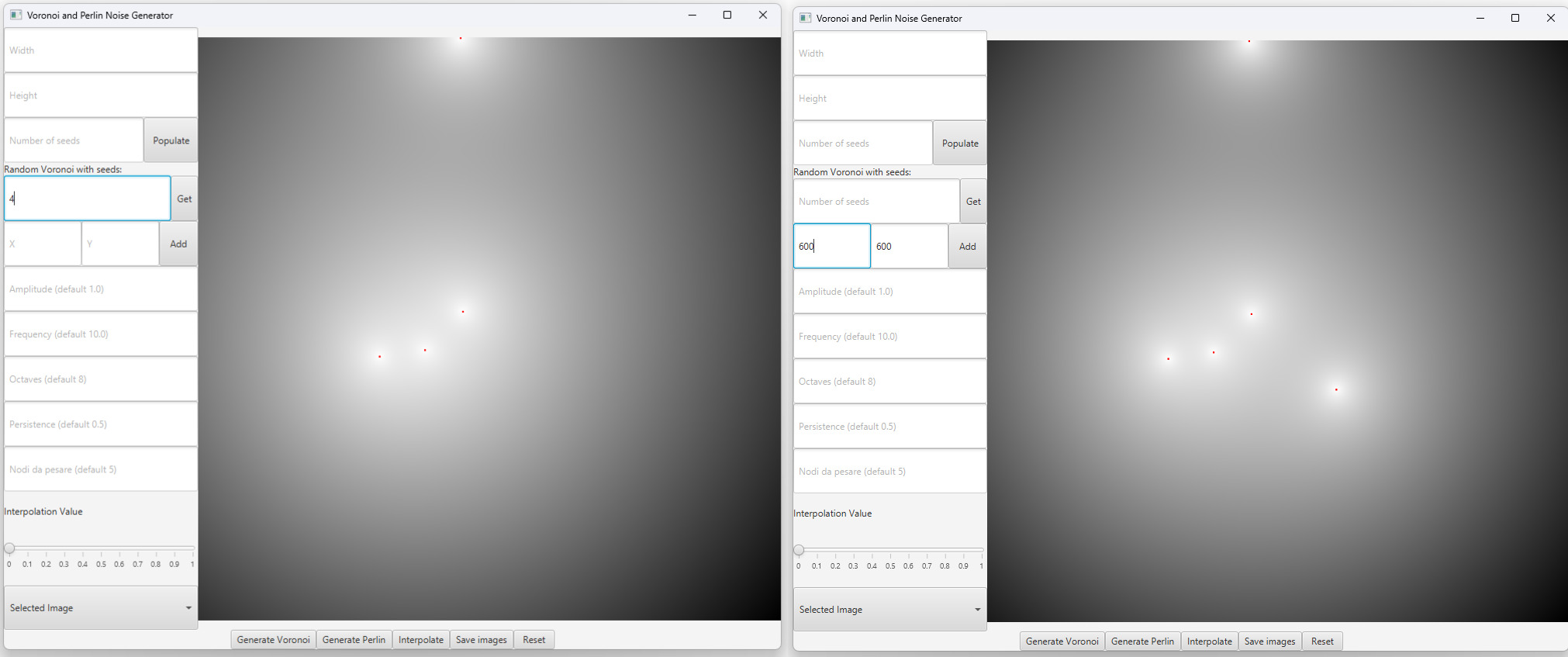
Un video esplicativo è disponibile al link: <https://youtu.be/u2JttjGjngk>

[](https://youtu.be/u2JttjGjngk)

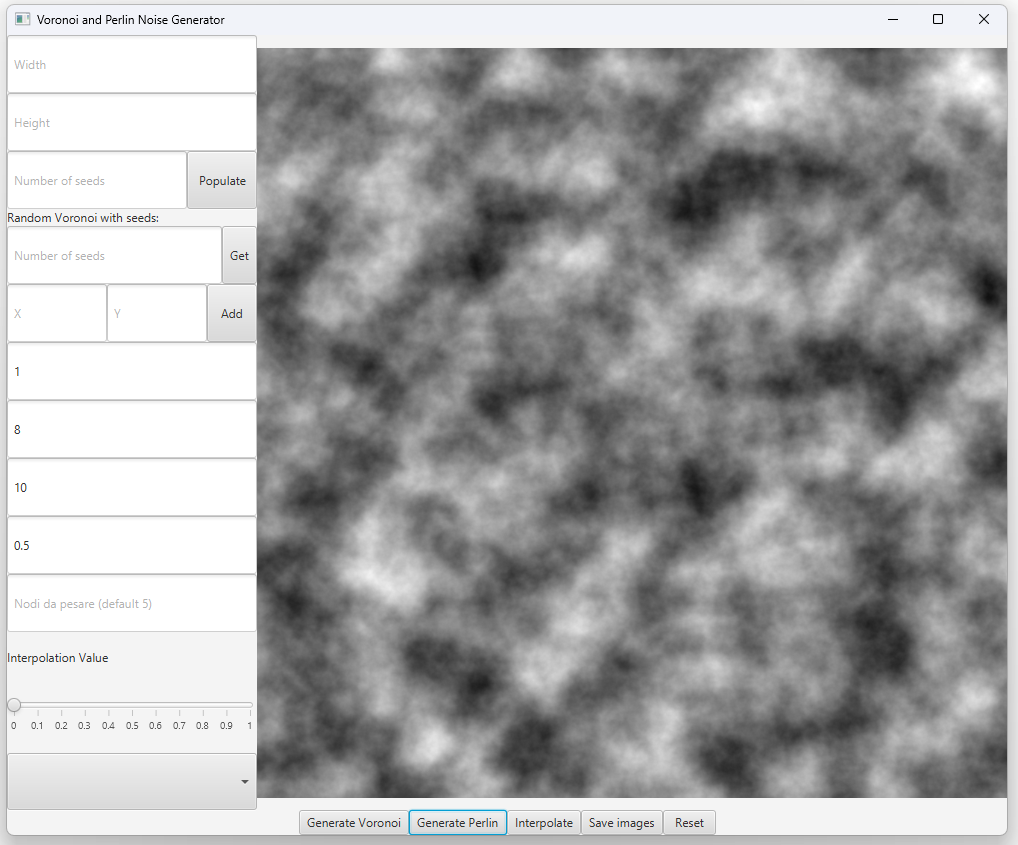
**Creazione Voronoi:**



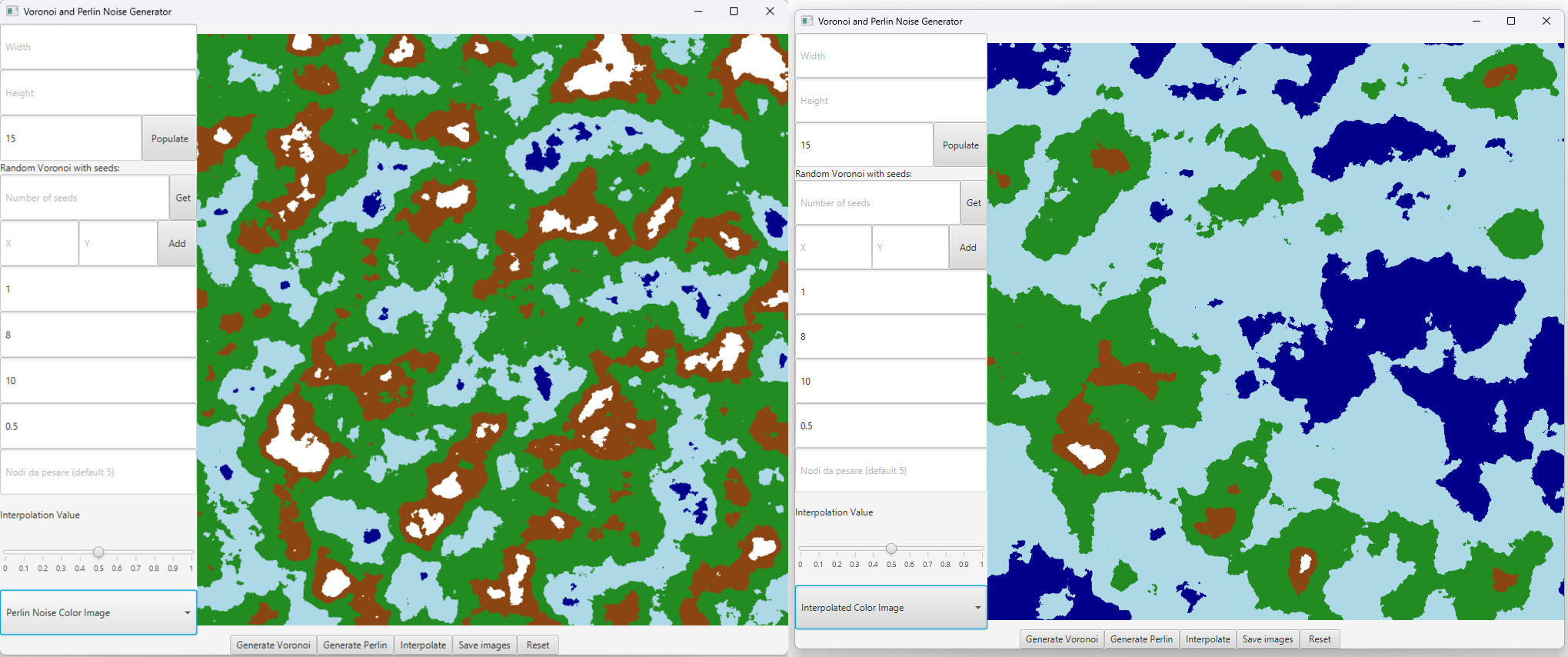
**Recupero da database e aggiunta nodo:**



**Creazione Perlin:**



**Interpolazione (a sinistra Perlin normale, a destra interpolato):**



**7 – Risultati**

Come possibile notare dall’ultima immagine il rumore di Perlin viene fortemente influenzato dal diagramma di Voronoi e i suoi nodi. Minore sarà il numero di nodi del diagramma, maggiore sarà diverso il risultato dal rumore iniziale, e viceversa. L’algoritmo funziona infatti meglio con minor numero di nodi, dato che all’aumentare del numero aumenta la loro distribuzione omogenea.

Questo algoritmo produce risultati soddisfacenti entro certi limiti dati dai parametri. Il risultato dell’interpolazione è molto meno omogeneo del rumore di Perlin originale, che era uno dei limiti dell’algoritmo originale.

Questo nuovo algoritmo può essere migliorato, integrando la possibilità di “avvolgere” la mappa intorno ai bordi (cosa richiesta per la generazione di textures) e rimuovendo la tendenza del diagramma di voronoi di avere (per base statistica) nodi riuniti verso il centro.

**Licenza**

Questa relazione è disponibile con licenza Creative Commons BY-NC-SA 4.0

Per maggiori informazioni visitare: https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode.it

