IB Mathematics Internal Assessment

MAP PROJECTIONS

Noise

June 8, 2025

Gia Phu Huynh

1 Introduction

Mathematics does not merely describe nature - it architects worlds. Within the intricate algorithm, virtual worlds take shape as jagged mountains rise, valleys unfurl, and rivers etch serpentine paths. This is not magic; it is mathematics in motion, bending stochasticity into artistry.

Traditionally, the primary technique used in developing terrain has been handcraft, using modelling software to precisely describe the creations. However, this approach suffers from four main drawbacks (Roden & Parberry, 2004):

- **Operability:** handcrafted terrains are slow and resource-intensive, requiring manual adjustments for every detail.
- **Inflexibility:** modifying completed designs can dramatically alter technical requirements.
- **Incompatibility:** handcrafted terrains often lack coherence with physics engines and other computational models.
- **Unscalability:** as virtual worlds grow in complexity with their expansive, dynamic environment, handcrafted terrains become impractical.

In the late 1960s, computer graphics pioneers confronted these setbacks, seeking ways to simulate natural patterns without the efficiencies of manual design (Autodesk, 2024). The answer manifested in the form of noise algorithms, mathematical functions capable of producing structured randomness, often termed "pseudo-randomness" (Perlin, 2001). Over the decades, they have evolved beyond gaming applications to enable hyper-realistic CGIs in films (Pegg, 2010), creating texture (Perlin, 1985), and advancing fluid dynamics simulations (Kim et al., 2008).

My fascination with noise algorithms began in the blocky worlds of the game Minecraft. As a teenager, I spent countless hours exploring its vast biomes, awestruck by the seamless blend of towering cliffs, dense forests, and sprawling cave systems. The terrain felt organic, but I knew it was generated algorithmically. While I appreciate the aesthetics of the game, I lacked insight into the technical requirements underpinning it. Years later, I discovered the role of noise in shaping these landscapes and became curious about which aspects of noise contribute to terrain realism.

There are two main types of noise: Value noise and Perlin noise. Despite the popularity, their ability to replicate real-world terrain has not been rigorously studied under controlled conditions. This IA will determine which algorithm produces more realistic terrain by statistically comparing

- Elevation distribution using **Histograms of normalised height values**.
- Spatial autocorrelation using **Moran's I to measure globl clustering trends**.
- Spectral characteristics using **Power Spectral Density to classify frequency dominance**.

with real-world elevation data from the Global Multi-resolution Terrain Elevation Data (GMTED2010), a model developed through a collaborative effort between the U.S. Geological Survey (USGS) and the National Geospatial-Intelligence Agency (NGA).

Terrain maps will be generated on a fixed 256×256 grid with normalised height values $z \in [0, 1]$. Both algorithms are tested under identical spectral parameters, including amplitude, frequency, persistence, lacunarity, and number of octaves. Cubic smoothstep functions will be applied to bilinear interpolations to enforce continuity. A fixed seed (30042603) will be employed to ensure deterministic comparisons by removing the randomness of the height values out of the effects.

Lastly, all implementations will be done using the Python programming language. All code will be available in the Appendix.

2 Noise Functions

2.1 Theory

Let $\mathcal{N} : \mathbb{R}^n \to \mathbb{R}$ denote the core noise function, where n is the dimensionality of the input domain. As mentioned in the introduction, there exist two variants: Value noise, $\mathcal{N}_V$, and Perlin noise, $\mathcal{N}_P$.

This IA specifically focuses on the two-dimensional case, where the noise generation algorithm operates on a pixel grid of resolution $N \times N$ iterating over integer coordinates $(u, v) \in \{0, 1, \ldots, N-1\}^2$. As established in the introduction, $N = 256$. These discrete coordinates are transformed into the normalised domain $Q_{\text{base}} = [0, 1)^2 \subset \mathbb{Q}^2$ via the transformation $(x, y) = \left(\frac{u}{N}, \frac{v}{N}\right)$. This ensures scale invariance, guaranteeing that relative feature positions remain constant under resolution change.

Let $G_V : \mathbb{Z}^2 \to [0, 1]$ and $\vec{G}_P : \mathbb{Z}^2 \to \mathbb{R}^2$ be lattice functions associated with Value noise and Perlin noise, respectively. For a domain $0 \leq i, j \leq N$, these functions map integer lattice coordinates to precomputed random values as follows:

- $G_V(i, j)$ returns a pseudorandom scalar $v_{i,j} \in [0, 1]$, sampled uniformly.
- $\vec{G}_P(i, j)$ returns a pseudorandom unit vector $\hat{v}_{i,j}$ sampled uniformly.

For any query point $(x, y) \in Q_{\text{base}}$, and let (i, j) be the coordinate of the top-left vertex associated with the chosen query coordinate, then because all vertex coordinates are integer pairs, the value of i and j can be found by rounding x and y down to the nearest integer, which is written as the floor function.

$$i = \lfloor x \rfloor, j = \lfloor y \rfloor$$

Subsequently, the associated top-right, bottom-right, and bottom-left vertices are respectively $(i+1, j)$, $(i+1, j+1)$, and $(i, j+1)$.

Finally, interpolate the query point with the values at the four lattice vertices to obtain the height value at the query point. Although bicubic and spline interpolations are possible, due to the limited scope of this IA, we will only be exploring bilinear interpolation with different polynomial smoothstep function $S_n(t)$, where n is the order of the function.

To derive the formula for bilinear interpolation, we first interpolate in the x-direction which yields two intermediate values

$$\begin{cases} \mathcal{N}(x, j) = \mathcal{N}(i, j) \cdot (1 - S_n(x - i)) + \mathcal{N}(i+1, j) \cdot S_n(x - i) \\ \mathcal{N}(x, j+1) = \mathcal{N}(i, j+1) \cdot (1 - S_n(x - i)) + \mathcal{N}(i+1, j+1) \cdot S_n(x - i) \end{cases}$$

Substitute $\mathcal{N}(i, j)$, $\mathcal{N}(i+1, j)$, $\mathcal{N}(i, j+1)$, and $\mathcal{N}(i+1, j+1)$ with $G_V(i, j)$, $G_V(i+1, j)$, $G_V(i, j+1)$, and $G_V(i+1, j+1)$ respectively

$$\begin{cases} \mathcal{N}(x, j) = G_V(i, j) \cdot (1 - S_n(x - i)) + G_V(i+1, j) \cdot S_n(x - i) \\ \mathcal{N}(x, j+1) = G_V(i, j+1) \cdot (1 - S_n(x - i)) + G_V(i+1, j+1) \cdot S_n(x - i) \end{cases}$$

We then interpolate in the y-direction, yielding our height value

$$\begin{aligned} \mathcal{N}_V(x, y) =& \mathcal{N}_V(x, j) \cdot (1 - S_n(y - j)) + \mathcal{N}_V(x, j+1) \cdot S_n(y - j) \\ =& [G_V(i, j) \cdot (1 - S_n(x - i)) + G_V(i+1, j) \cdot S_n(x - i)] \cdot (1 - S_n(y - j)) \\ &+ [G_V(i, j+1) \cdot (1 - S_n(x - i)) + G_V(i+1, j+1) \cdot S_n(x - i)] \cdot S_n(y - j) \end{aligned}$$

which can be condensed into the matrix expression

$$\mathcal{N}_V(x, y) = \begin{bmatrix} 1 - S_n(x - i) & S_n(x - i) \end{bmatrix} \begin{bmatrix} G_V(i, j) & G_V(i+1, j) \\ G_V(i, j+1) & G_V(i+1, j+1) \end{bmatrix} \begin{bmatrix} 1 - S_n(y - j) \\ S_n(y - j) \end{bmatrix} \tag{1}$$

Equation 1 can be modified to suit $\mathcal{N}_P(x, y)$ by replacing $G_V(i, j)$ with the dot product $g_{i,j} = \vec{G}_P(i, j) \cdot \vec{r}_{i,j}$, where $\vec{r}_{i,j}$ is the displacement vector from point (x, y) to (i, j).

To generate a full terrain, repeat the above steps for all query points (x, y).

2.2 Key Parameters

To control the characteristics of a noise function, we can introduce two adjustable parameters: frequency and amplitude. Frequency, $0 < \lambda \leq \frac{N}{2}$, determines the density of features within the terrain - greater frequency leads to more granular details. The upper bound on frequency derives from the Nyquist-Shannon sampling theorem, which, according to Candes and Wakin (2008), states that "the sampling rate must be at least twice the maximum frequency present in the signal." Consequently, having integrated frequency into account, the effective domain of the noise function now becomes $Q = [0, \lambda)^2$. Amplitude, $A > 0$, defines the overall magnitude of the noise function - where $A > 1$ lead to greater extremes in the terrain elevation while $0 < A < 1$ leads to less extremes.

Incorporating both parameters, we modify the noise function as follows

$$A \cdot \mathcal{N}(\lambda x, \lambda y)$$

Next, recall that the smoothstep function $S_n(t)$ serves as an interpolation weight in noise functions, as seen in Equation 1. It must satisfy the following conditions:

- **Boundary conditions:** $S_n(0) = 0$ and $S_n(1) = 1$.
- **Continuity:** for $m = 1, 2, \ldots, \lfloor \frac{n}{2} \rfloor$, the m^{th} derivative of S_n satisfies $S_n^{(m)}(0) = 0$ and $S_n^{(m)}(1) = 0$. The floor function ensures m is an integer, as fractional derivatives are not considered here.
- **Monotonicity** (strictly increasing): $S_n'(t) > 0, \forall t \in (0, 1)$
- **Rotational symmetry about** $(0.5, 0.5)$**:** This condition can be derived by first defining a shifted function that centers the point of symmetry at the origin:

  $$f(u) = S_n\left(u + \frac{1}{2}\right) - \frac{1}{2}$$

  then enforcing conditions of an odd function to enforce symmetry:

  $$f(-u) = -f(u)$$
  $$S_n\left(-u + \frac{1}{2}\right) - \frac{1}{2} = -\left[S_n\left(u + \frac{1}{2}\right) - \frac{1}{2}\right]$$

  Finally, substituting $t = \frac{1}{2} - u$, then simplify:

  $$S_n(t) - \frac{1}{2} = -\left[S_n(1 - t) - \frac{1}{2}\right]$$
  $$S_n(t) = 1 - S_n(1 - t)$$

Having established the conditions, we will now attempt to construct a smoothstep function by first giving it the general form of a polynomial

$$S_n(t) = \sum_{k=0}^{n} a_k t^k$$

where a_k is the coefficient of the k^{th} term.

From the symmetry condition, it follows that n must be odd (Wikimedia Foundation, 2025). Although I attempted to formally prove this result, I was unable to reach a conclusive one. Therefore, for the purposes of this Internal Assessment, this property will be assumed to hold.

From the continuity condition $S_n^{(m)}(1) = 0$, we know that there is no constant term in any of the derivative up to $m = \lfloor \frac{n}{2} \rfloor$, hence the lowest-order term in the polynomial $S_n(t)$ is $t^{\lfloor \frac{n}{2} \rfloor + 1}$. This is because each term in the polynomial contributes a non-zero constant to the m^{th} derivative at $t = 1$ when $k \leq m$. By eliminating all terms where $k \leq \lfloor \frac{1}{2} \rfloor$, we ensure that every required derivative automatically vanishes at the endpoint. Amending this finding, we rewrite the smoothstep function as

$$S_n(t) = \sum_{k=\lfloor \frac{n}{2} \rfloor + 1}^{n} a_k t^k$$

We now see another benefit of the floor function for $\frac{n}{2}$, which is to ensure that the index k in the sigma summation is always an integer. We might have also noticed that there is no upper-bound for n, which means that infinitely many smoothstep functions are available. For this investigation, I limit consideration to cubic and quintic polynomials, as higher-degree polynomials will introduce too many coefficients to solve for, as well as costing computation time.

With all the conditions and boundaries set, we aim to analytically solve for $S_3(t)$ defined as:

$$S_3(t) = a_2t^2 + a_3t^3$$

Applying the boundary condition and continuity condition, we obtain the simultaneous equation:

$$\begin{cases} \sum_{k=2}^{3} a_k = 1 \\ S_3'(1) = 0 \end{cases} \Rightarrow \begin{cases} a_2 + a_3 = 1 & \text{Ⓐ} \\ 2a_2 + 3a_3 = 0 & \text{Ⓑ} \end{cases} \tag{2}$$

Rewrite equation 2A in terms of a_3

$$a_3 = 1 - a_2$$

then substitute into equation 2B

$$\begin{aligned} 2a_2 + 3(1 - a_2) &= 0 \\ 2a_2 + 3 - 3a_2 &= 0 \\ a_2 &= 3 \end{aligned}$$

Now, substitute the value of a_2 back into equation 2A

$$a_3 = 1 - (3) = -2$$

Thus, the smoothstep function of order 3 becomes:

$$S_3(t) = -2t^3 + 3t^2 \tag{3}$$

As mentioned, we will also attempt to derive the quintic smoothstep function $S_5(t)$

$$S_5(t) = a_3t^3 + a_4t^4 + a_5t^5$$

Applying the continuity and boundary condition:

$$\begin{cases} \sum_{k=3}^{5} a_k = 1 \\ S_5'(1) = 0 \\ S_5''(1) = 0 \end{cases} \Rightarrow \begin{cases} a_3 + a_4 + a_5 = 1 \\ 3a_3 + 4a_4 + 5a_5 = 0 \\ 6a_3 + 12a_4 + 20a_5 = 0 \end{cases} \Rightarrow \begin{cases} a_3 + a_4 + a_5 = 1 \\ 3a_3 + 4a_4 + 5a_5 = 0 \\ 3a_3 + 6a_4 + 10a_5 = 0 \end{cases} \tag{4}$$

PUT GAUSSIAN ELIMINATION STUFF HERE LATER

There are two other parameters we will adress: persistence and lacunarity. Typically, a noise function with a single frequency will look overly smooth and artificial because it only captures one level of detail, as illustrated in [FIGURE REF] and [FIGURE REF] in [SECTION REF]. Natural terrains, exhibit complexity at multiple scales. For instance, a mountain presents broad overall forms alongside finer features such as ridges and crevices layered atop one another.

To replicate the complexity of natural terrains, fractal noise is used where multiple layers of noise (called octaves) are summed, each with decreasing amplitude and increasing frequency. The parameter persistence, $0 < p < 1$, controls the rate at which amplitude decay, ensuring higher frequency details do not overwhelm the overall coherence. Lacunarity, $L > 1$, determines the rate of frequency growth, introducing finer-scale variations. Finally, the total number of octaves, $C \geq 1$, controls the level of detail in the terrain. Putting this together, the final terrain height at pixel (u, v) is given by:

$$\mathcal{N}_{FV}(x, y) = \sum_{k=0}^{C-1} A_k \cdot \mathcal{N}_V\left(\lambda_k \frac{u}{N}, \lambda_k \frac{v}{N}\right)$$

where

- $A_k = A_0 \cdot P^k$
- $\lambda_k = \lambda_0 \cdot L^k$

$\mathcal{N}_V$ can be replaced by $\mathcal{N}_P$, and thus $\mathcal{N}_{FV}(x, y)$ to $\mathcal{N}_{FP}(x, y)$, depending on which noise variant is being used. One final detail is determining the upper bound for C. Recall that λ_{C-1} must satisfy the Nyquist limit mentioned earlier, which gives us the inequality:

$$\begin{aligned} \lambda_{C-1} &\leq \frac{N}{2} \\ \lambda_0 \cdot L^{C-1} &\leq \frac{N}{2} \\ L^{C-1} &\leq \frac{N}{2\lambda_0} \\ C &\leq 1 + \log_L \left(\frac{N}{2\lambda_0} \right) \end{aligned}$$

Having introduced the core theory and parameters in this IA, I will now present the quantitative tools that will be used to compare the noise functions.

2.3 Quantitative Metrics

Firstly, all elevation values generated from noise functions are aggregated and plotted onto a histogram to analyze their statistical distribution. The number of intervals were determined using Rice rule, a preferred method which "set the number of intervals to twice the cube root of the number of observations." (Lane & Scott, n.d.). For our dataset of $256 \times 256 = 65536$ samples, this yielded 81 intervals after rounding up. To further analyze the elevation values generated from the noise function, several statistical measures were computed to capture various characteristics of the terrain.

Let $X_{u,v}$ defines the elevation value at pixel (u, v), the mean (μ) provides the average elevation, giving a sense of the overall altitude of the terrain, and is calculated using the equation:

$$\mu = \frac{1}{N^2} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} X_{u,v}$$

The variance (σ^2) quantifies how spread out the elevations are from the mean, where a higher variance indicates a more varied terrain:

$$\sigma^2 = \frac{1}{N^2} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} (X_{u,v} - \mu)^2$$

To assess the shape of the distribution, skewness (γ_1) is used to measure asymmetry in the dataset, where a value close to 0 suggests a symmetric distribution of heights.

$$\gamma_1 = \frac{1}{N^2} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \left(\frac{X_{u,v} - \mu}{\sigma} \right)^3$$

Furthermore, kurtosis, precisely Fisher's kurtosis, will be used to measure the "peakedness" of the elevation distribution compared to a normal distribution, where positive kurtosis means a more peaked distribution, while negative kurtosis means a flatter one.

$$\gamma_2 = -3 + \frac{1}{N^2} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \left(\frac{X_{u,v} - \mu}{\sigma} \right)^4$$

Finally, the Terrain Ruggedness Index (TRI) quantifies how much the elevation changes between a grid cell and its neighbors, with higher TRI values indicate rougher terrains.

$$\text{Mean TRI} = \frac{1}{(N-2)^2} \sum_{u=1}^{N-2} \sum_{v=1}^{N-2} \text{TRI}_{u,v}$$

with

$$\mathrm{TRI}_{u,v} = \sqrt{\sum_{(a,b)\in\mathcal{P}} (X_{u,v} - X_{u+a,v+b})^2}$$

where

$$\mathcal{P} = \{(-1,-1),(-1,0),(-1,1),(0,-1),(0,1),(1,-1),(1,0),(1,1)\}$$

All of these metrics will be computed using technology (see APPENDIX REF).

Another quantitative metric I will use is spatial autocorrelation which is often used to describe the degree to which a variable is correlated with itself across space (Moraga, 2023). Positive spatial autocorrelation indicates that similar values are clustered whereas negative spatial autocorrelation indicates that they are dispersed.

One method, proposed by P. A. P. Moran in 1950, to assess spatial autocorrelation is Moran's I coefficient (Moran, 1950). Although R. C. Geary also proposed his Geary's C coefficient (Geary, 1954), it focuses on local dissimilarities rather than overall similarities, which is not what my IA wanted to focus on. On that note, Moran's I takes the form

$$I = \frac{N}{W} \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} w_{i,j}(x_i-\mu)(x_j-\mu)}{\sum_{i=1}^{N}(x_i-\mu)^2}$$

where

- N is the number of variables indexed by i and j;
- x is the variable of interest;
- μ is the mean;
- $w_{i,j}$ are the elements of a spatial weight matrix with zeroes on the diagonal;
- $W = \sum_{i=1}^{N}\sum_{j=1}^{N} w_{i,j}$.

[FIGURE INSERT HERE]

An inverse Euclidean distance weight matrix was decided with a 16 pixels radius cutoff as it penalizes distant pixels so key structural differences between the two noise types can be accurately captured. The formula is:

$$w_{i,j} = \begin{cases} \frac{1}{d_{i,j}}, & \text{if } d_{i,j} \le 16 \\ 0, & \text{otherwise} \end{cases} \qquad \text{where } d_{i,j} \text{ is the distance between pixel indexed } i \text{ and } j$$

It is worth noting that here, the index i and j relates to pixel coordinates with the formula $i = u + 256v$. Inversely, $u = i \mod 256$ and $v = \lfloor \frac{i}{256} \rfloor$. The complete spatial weight matrix, and thus the Moran's I coefficient will be computed using technology (see Appendix REF).

The final quantitative metric I will use is called the power spectral density, a potent tool in signal processing which quantifies the power distribution of a signal across frequencies (Slavič et al., 2021, p.64). For our use case, where the signal is discrete and two dimensional, the power spectral density can be expressed as (Blackman & Tukey, 1958, p.193):

$$P(u,v) = \frac{1}{MN}\|\mathcal{F}(u,v)\|^2$$

where $\mathcal{F}(u,v)$ is the 2-D Discrete Fourier Transform (DFT) (Gonzalez & Woods, 2007, p.257):

$$\mathcal{F}(u,v) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} f(x,y)e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

Adapting to this IA, we will firstly substitute $M = N$ due to our square terrain map and secondly, substitute $f(x,y)$ with $\mathcal{N}_{FV}(x,y)$ or $\mathcal{N}_{FP}(x,y)$.

After calculating for all values of u and v, plot $\log P(u,v)$ onto the pixel (u,v). Due to the computation-heavy nature of this algorithm, the PSD will be done using technology (see Appendix REF).

3 Results

3.1 Calculation

Before using technology to produce a complete terrain map, I will show the hand calculation for one pixel $(u, v) = (3, 143)$ in both Value and Perlin noise. The parameters will be $\lambda_0 = 2$, $A_0 = 1$, $L = 2$, $P = 0.5$, and $O = 6$. The seed, as mentioned in the introduction, is 30042603.

Firstly, transform pixel coordinates into the domain of the noise function, then scale by λ_0 which gives:

$$\left(\frac{u}{N}\lambda_0, \frac{v}{N}\lambda_0\right) = \left(\frac{3}{256} \cdot 2, \frac{143}{256} \cdot 2\right) = \left(\frac{3}{128}, \frac{143}{128}\right)$$

Next, find the associated top-left vertex:

$$(i, j) = \left(\left\lfloor \frac{3}{128} \right\rfloor, \left\lfloor \frac{143}{128} \right\rfloor\right) = (0, 1)$$

which means the other three vertices are $(1, 1)$, $(0, 2)$, and $(1, 2)$.

Generating random scalar values for the lattice function G_V with the seed, we obtain:

$$\begin{bmatrix} G_V(0,1) & G_V(1,1) \\ G_V(0,2) & G_V(1,2) \end{bmatrix} = \begin{bmatrix} 0. & 0. \\ 0. & 0. \end{bmatrix}$$

Substituting these values into our bilinear interpolation (Equation 1), with the cubic smoothstep function $S_3(t)$ (Equation 3):

$$\begin{aligned} \mathcal{N}_V\left(\frac{3}{128}, \frac{143}{128}\right) &= \begin{bmatrix} 1 - S_n(\frac{3}{128} - 0) & S_n(\frac{3}{128} - 0) \end{bmatrix} \begin{bmatrix} G_V(0,1) & G_V(1,1) \\ G_V(0,2) & G_V(1,2) \end{bmatrix} \begin{bmatrix} 1 - S_n(\frac{143}{128} - 1) \\ S_n(\frac{143}{128} - 1) \end{bmatrix} \\ &\approx a \end{aligned}$$

Finally, multiply the value by the amplitude to get the final elevation value at the pixel $(3, 143)$.

$$A_0 \cdot \mathcal{N}_V\left(\frac{3}{128}, \frac{143}{128}\right)$$

Repeat this process and plot them to all the appropriate pixels using technology, we will obtain the full map for one set of amplitude and frequency.

3.2 Visual Comparison

3.3 Quantitative Comparison

4 Evaluation

While the outcome of this investigation was rather successful, there were notable assumptions and methodological limitations that could have been better addressed and improved.

One of the most significant setbacks was the initial decision to generate terrain maps at a resolution 1024×1042 pixels. While the intention was to match the granularity of real-world terrain, this approach quickly proved computationally efficient, especially for the Moran's I calculations which took at least 2 hours to finish for one map. To ensure feasibility within the timeline, I had to reduce the resolution to 256×256. Although this allowed for smoother experimentation, it introduced the trade off that the lower resolution may not have captured finer-scale topographic features, potentially limiting the fairness of direct comparisons with real-world datasets. Even when I tried to resample real-world datasets to match the 256×256 resolution of the noise maps, it smoothed and altered the fine-scale features in the original terrain data, which downgraded the accuracy of spatial and spectral comparisons. In future investigations, a multiscale analysis could be adopted to examine noise performance at multiple resolutions.

Moreover, several assumptions were made without rigorous justification. For instance, while constructing the smoothstep functions used in interpolation, I assumed that the order must be odd to satisfy a symmetry condition. Although this assumption aligns with some sources, I was unable to produce a formal mathematical proof. This is an area where greater rigor could be applied in future iterations of this investigation.

A similar limitation concern the preference of Moran's I over Geary's C in evaluating spatial autocorrelation. While I justified this based on the focus of my analysis on global similarity patterns, I did not conduct a comparative analysis of the two metrics, nor did I examine cases where Geary's C might yield extra helpful insights. Including such a comparison could have added depth to my conclusions and strengthened the reliability of the findings.

Another methodological refinement that could enhance the evaluation of realism is the classification of noise colour. Terrain surfaces, like many natural phenomena, often exhibit red noise characteristics, where lower spatial frequencies dominate. Incorporating spectral slope analysis to determine whether the noise resembles red, pink, white, or blue noise would offer an additional dimension of realism assessment. Since human perception tends to associate "naturalness" with red noise (Haroz, 2006), this framework could help formalize qualitative judgments of visual realism.

I had also utilized only one set of parameters to perform the comparison, which discards numerous other possibilities that could have enhanced the synthetic noise similarities to real terrain. This was done because otherwise it would go beyond the scope of this IA. However, in the future, I would investigate with different parametric settings; although I cannot include all in the paper, at least I could compile into a concise table and put in the Appendix.

Finally, a limitation inherent in my methodology was the reliance on a fixed seed for initial comparisons. Larger sample size could have been employed to more robustly quantify the variance and confidence in the conclusion.

5 Conclusion

In conclusion, I successfully achieved my aim of the investigation and demonstrated the superiority of the Perlin noise function in simulating realistic terrain features. This was accomplished through a comprehensive evaluation based on both statistical measures, such as the Moran's I spatial autocorrelation, and spectral characteristics.

In terms of basic elevation statistics, Perlin noise exhibits a higher mean elevation and greater variance than Value noise. These metrics align more closely with natural landscapes, suggesting that Perlin noise better captures the moderate elevation shifts observed in alpine topography. Its negative kurtosis further mirrors that of real terrains like Everest and Matterhorn, indicating a flatter distribution with fewer extreme outliers. Although Value noise shows a favorable skewness profile, this similarity appears to be an artifact of randomness.

The power spectral density (PSD) analysis provides further evidence in favor of Perlin noise. Its frequency domain representation reveals a strong central energy concentration and a gradual decay towards higher frequencies, resembling the spectral patterns observed in terrains such as Everest and Denali. In contrast, Value noise produces a flatter, more uniform PSD. The absence of distinct low-frequency dominance and directional variation implies a lack of structural realism.

However, both synthetic models remain fundamentally limited in capturing the true complexity of real-world landscapes. Not only did both fail to capture the full statistical range observed in natural terrain, most notably the variance metric, its PSD stray far from resembling the frequency decay observed in natural terrain.

This inquiry has significantly deepened my understanding of noise functions, particularly how their underlying mathematical foundations and algorithms influence their capacity to replicate the complexity and variability found in natural landscapes. Additionally, the research process broadened my knowledge with statistical tools and introduced me to spectral analysis and signal processing techniques, most notably the use of Power Spectral Density, which I had not encountered before.

Given that this area of study remains relatively underexplored, with limited literature due to the high competition between game companies, I often found myself navigating through a labyrinth with few established guidelines. While this was challenging, the lack of precedent also gave me the freedom to formulate and test original approaches. I am particularly proud of the fact that many of the techniques and comparisons I employed were conceptualized on my own. This experience serves more than just a stepping-stone for my problem-solving and analytical abilities but also affirmed my enthusiasm for mathematical research.

Looking ahead, I am eager to further explore this topic at the university level, where I can apply more advanced and rigorous mathematical tools and contribute to a promising field with fresh insights and greater technical depth.