Noise Input Parameters (Universal Reference)

A unified **.md reference file** for all types of gradient, value, simplex, and cellular noise.
Use this as a standard input specification across your .NET C# noise library.

📌Universal Noise Input Parameters

These parameters apply broadly to:
Perlin, Improved Perlin, Simplex, OpenSimplex, Value Noise, Worley (Cellular), and hybrid noises.

---

1. Core Sampling Parameters

`x`, `y` (float)

- Input coordinates to evaluate noise at.
- Can be fractional.
- Interpretation depends on scale and domain.

Example:

```
float value = Noise2D.Perlin(x, y);
```

---

2. `frequency` (float)

Controls the spatial scale of the noise. - Higher = more detail (smaller features) - Lower = smoother, larger structures

Default: `1.0f`

Formula effect:

```
noise(x, y, frequency) = baseNoise(x * frequency, y * frequency)
```

---

3. `seed` (int)

Determines the deterministic randomness of gradients/features. - Same seed = same results - Different seed = new noise pattern

Used internally to build: - Permutation table (Perlin/Simplex) - Feature point hashing (Worley)

🕑Fractal (fBm) Parameters

Used for **Perlin/Simplex/OpenSimplex/Value** fractal sums.

4. `octaves` (int)

Number of layers of noise to sum. - More octaves = richer detail - Too many = slower performance

Typical: `4–8`

---

5. `persistence` (float)

Controls amplitude decrease per octave.

Used in fBm:

```
amplitude *= persistence;
```

Common value: `0.4 – 0.6`

---

6. `lacunarity` (float)

Controls frequency increase per octave.

Used in fBm:

```
frequency *= lacunarity;
```

Common value: `2.0`

📦 Interpolation Parameters (for Value & Hybrid Noise)

7. `interpolation` (enum)

Controls the smoothing function used.

Modes:

- `Linear` — fast, blocky
- `Smoothstep` — cubic smoothing
- `Quintic` — standard fade curve

🧹 Cellular (Worley) Parameters

Used only for **Worley/Cellular Noise** or hybrid blending.

---

8. `density` (float)

Feature points per cell. - Higher density = smaller cells

---

9. `distanceMetric` (enum)

Determines how distances are computed.

Options: - `Euclidean` - `Manhattan` - `Chebyshev`

---

10. `jitter` (float)

Adds randomness to feature point positions.

- `0.0` → grid-like cells
- `1.0` → full jitter (organically shaped cells)

🕐 Domain Transform Parameters (Advanced)

Optional transform parameters commonly used in procedural generation.

---

11. `offsetX`, `offsetY`

Offsets applied **before sampling** to pan the noise.

---

12. `rotation` (float)

Angle (radians) to rotate the sampling domain. Used for motion effects, turbulence, and warping.

---

13. `warpStrength` (float)

Used for **domain warping** (dynamic distortion effects):

```
float dx = Noise(x * f1, y * f1) * warpStrength;
float dy = Noise(x * f1, y * f1) * warpStrength;
return Noise(x + dx, y + dy);
```

🕐 Output Formatting Parameters

14. `normalize` (bool)

When `true`, remaps output to `[0,1]`.

Formula:

```
(v + 1) * 0.5
```

---

15. `clamp` (bool)

Clamps final output to a range.

```
Math.Clamp(value, min, max)
```

🧱 Summary Table

Parameter	Type	Use	Applies To
`x, y`	float	Sample position	All noise types
`frequency`	float	Scale	All
`seed`	int	Deterministic randomness	All
`octaves`	int	Layers	Perlin, Simplex, Value
`persistence`	float	Amplitude decay	fBm
`lacunarity`	float	Frequency growth	fBm
`interpolation`	enum	Smoothing	Value + Hybrid
`density`	float	Feature points	Worley
`distanceMetric`	enum	Metric	Worley
`jitter`	float	Random feature offset	Worley
`offsetX/Y`	float	Position transform	All
`rotation`	float	Domain transform	All
`warpStrength`	float	Domain warping	All
`normalize`	bool	Map [-1,1] → [0,1]	All
`clamp`	bool	Range control	All

🔧 Example Unified C# Method Signature

```
public static float Noise(
    float x,
    float y,
    int seed = 0,
    float frequency = 1f,
    int octaves = 4,
    float persistence = 0.5f,
    float lacunarity = 2f,
```

```
    NoiseType type = NoiseType.Perlin,
    Interpolation interp = Interpolation.Quintic,
    float density = 1f,
    DistanceMetric metric = DistanceMetric.Euclidean,
    float jitter = 1f,
    bool normalize = false
)
```

If you want, I can now create: - A **second .md file** listing *all formulas only*, or - A **third .md file** with *visual diagrams* (ASCII or image-ready), or - Full **C# code templates** for each noise type.

Just tell me what you want next!