

Noise Testing Guide (For .NET C# Projects)

A complete **test plan + verification steps** for all noise algorithms: Perlin, Improved Perlin, Simplex, OpenSimplex, Value Noise, Worley, and hybrid noise functions. This file is meant to be included in your repository as a reference and checklist.

1. Purpose of This Test Suite

This document ensures that your noise implementation: - Produces **deterministic outputs** for a given seed. - Returns values in correct numeric **ranges**. - Shows no visible **artifacts**. - Supports **fractal parameters** (octaves, lacunarity, persistence). - Handles **domain transformations** correctly. - Performs well under large sampling grids.

2. Testing Structure

We test noise using **three categories**: 1. **Functional Tests** – correctness of math and outputs. 2. **Visual Tests** – generate image heatmaps to check artifacts. 3. **Performance Tests** – stress tests on CPU.

3. Functional Tests

These tests validate all noise functions.

3.1 Deterministic Output Test

Same coordinates + same seed = exact same output.

Expected Behavior: `noise(x,y,seed)` must equal repeated calls.

Test:

```
float a = Noise2D.Perlin(10.25f, 8.75f, seed: 99);  
float b = Noise2D.Perlin(10.25f, 8.75f, seed: 99);  
Debug.Assert(a == b);
```

Repeat for all noise types.

3.2 Range Test

All gradient/value noise must return `[-1, 1]` before normalization.

```
float v = Noise2D.Simplex(x, y);  
Debug.Assert(v >= -1f && v <= 1f);
```

For Worley noise, test `≥ 0` but no strict upper bound.

3.3 Frequency Test

When frequency doubles, features must become finer.

Procedure: 1. Sample rows of noise at frequency = 1 2. Repeat at frequency = 2 3. Compare variance:

```
Var(freq2) > Var(freq1)
```

3.4 Fractal Noise Test

Check that adding octaves increases detail.

```
float baseVal = Noise2D.PerlinFractal(8, 8, octaves: 1);  
float detailVal = Noise2D.PerlinFractal(8, 8, octaves: 6);  
Debug.Assert(baseVal != detailVal);
```

3.5 Interpolation Test

Only for value/hybrid noise.

- Linear = blocky
- Smoothstep = smoother
- Quintic = smoothest

Test by comparing the derivative: quintic should have smallest slope changes.

3.6 Seed Variation Test

Different seeds must produce different patterns.



4. Visual Tests

Visual inspection is crucial because noise quality is seen, not just measured.

4.1 Heatmap Output

Generate a `512×512` bitmap using each noise type. Check for: - Grid artifacts (bad in classic Perlin) - Diagonal/pattern noise - Sudden jumps or discontinuities - Wrong frequency distribution

4.2 Octave Heatmaps

Generate separate images for octaves 1–6. Ensure: - Higher octaves show finer details - No ringing artifacts

4.3 Domain Warp Test

Apply domain warping & visualize. Check if: - Distortion looks smooth - No clipping



5. Performance Tests

Performance matters in realtime applications.

5.1 Large Grid Test

Compute a `1000×1000` grid. Measure time:

```
< 50 ms → Excellent
< 120 ms → Good
< 250 ms → Acceptable
> 300 ms → Needs optimization
```

5.2 Multi-Octave Stress Test

Sample `10,000` fractal noise values with 8+ octaves. Ensure no GC allocations occur in repeated calls.



6. Integration Tests

Used when noise drives terrain, clouds, textures, etc.

6.1 Terrain Heightmap Test

Noise should: - Generate smooth hills - Avoid repeated tile patterns

6.2 Animation Coherence Test

For 3D noise or time-evolving noise: - Time slices must change smoothly - No flickering



7. Automated Testing in C# (NUnit Example)

```
[Test]
public void Perlin_IsDeterministic()
{
    float a = Noise2D.Perlin(3.4f, 7.8f, seed: 5);
    float b = Noise2D.Perlin(3.4f, 7.8f, seed: 5);
    Assert.AreEqual(a, b);
}

[Test]
public void Simplex_ReturnsRange()
{
    float v = Noise2D.Simplex(12.6f, -8.12f);
    Assert.IsTrue(v >= -1f && v <= 1f);
}
```



8. Folder Structure Suggestion

```
/Noise
  Noise2D.cs
  Perlin.cs
  Simplex.cs
  Worley.cs

/Tests
  PerlinTests.cs
```

```
SimplexTests.cs  
WorleyTests.cs  
VisualTests.cs
```

```
/Docs
```

```
Noise_Testing_Guide.md  
Noise_Input_Parameters.md  
GradientNoise_Types_and_Implementation.md
```

Final Notes

This test suite: - Ensures mathematical correctness - Gives visual certainty of quality - Guarantees deterministic, reproducible results - Helps detect artifacts early

If you want, I can also generate: - **Unit test C# boilerplate files** - **BenchmarkDotNet performance tests** - **Script to auto-generate heatmaps**