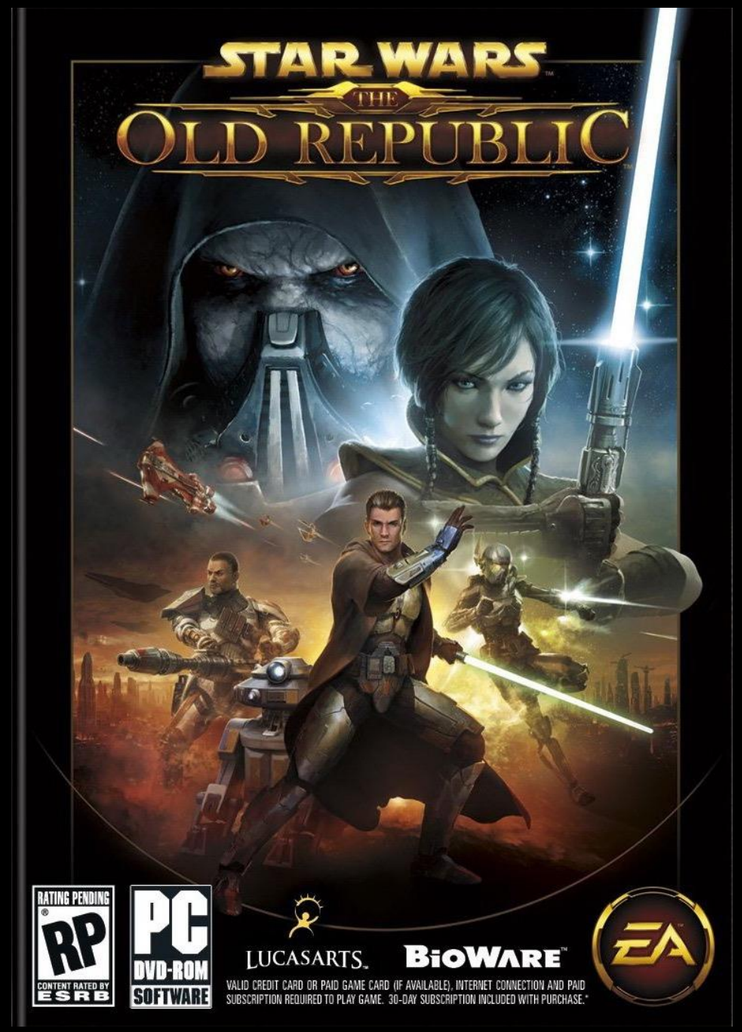
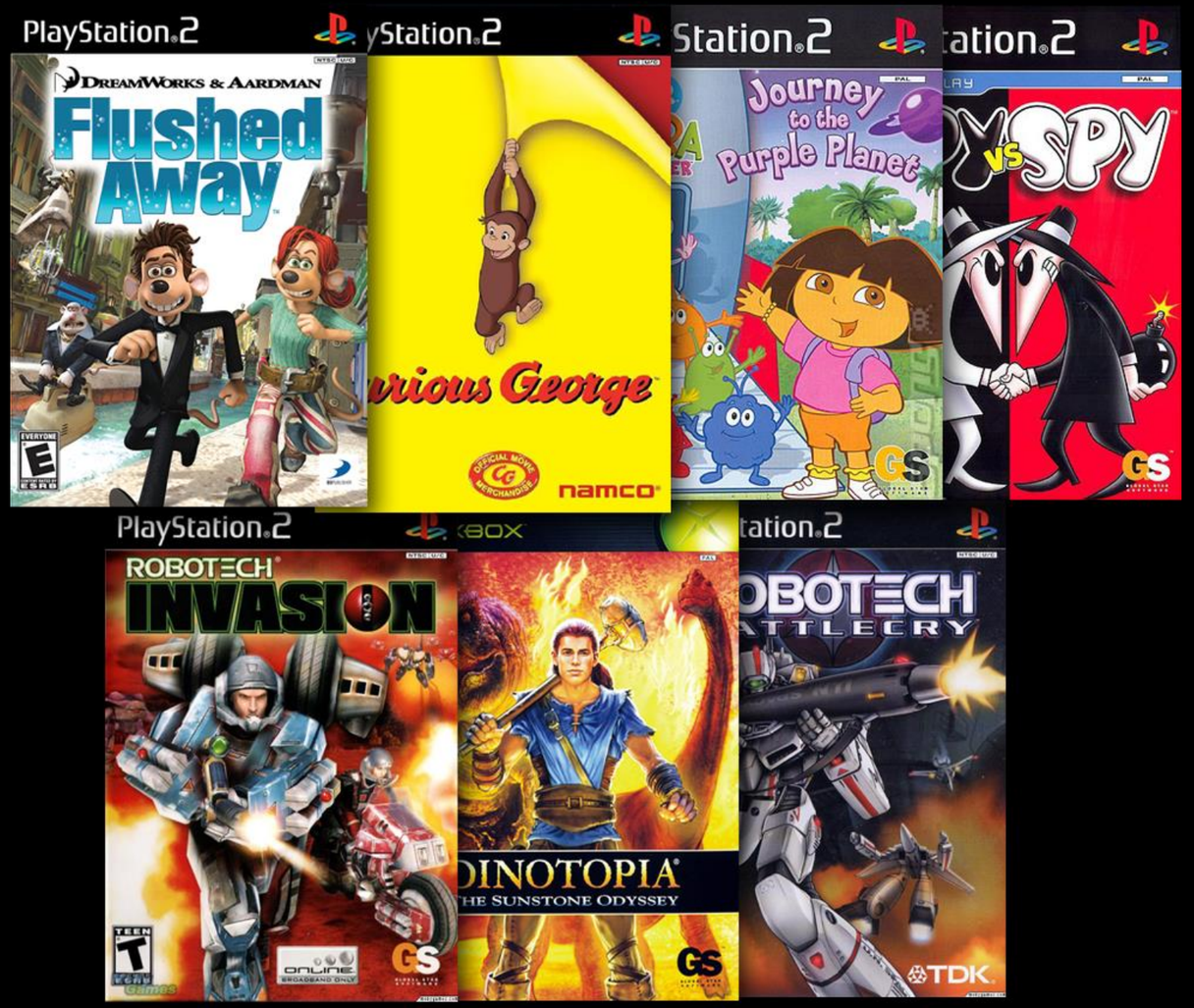
Shaders 301

Ben Cloward
CG Supervisor
Bioware

Who Are You?


STAR WARS
THE
OLD REPUBLIC
RATING PENDING
RP
CONTENT RATED BY
ESRB
PC
DVD-ROM
SOFTWARE
LUCASARTS
BIOWARE
EA
VALID CREDIT CARD OR PAID GAME CARD (IF AVAILABLE), INTERNET CONNECTION AND PAID SUBSCRIPTION REQUIRED TO PLAY GAME. 30-DAY SUBSCRIPTION INCLUDED WITH PURCHASE.*


PlayStation.2
DREAMWORKS & AARDMAN
Flushed Away
EVERYONE
E
ESRB
PlayStation.2
namco
Station.2
Journey to the Purple Planet
GS
tation.2
SPY vs SPY
GS
PlayStation.2
XBOX
tation.2
ROBOTECH
INVASION
TEEN
T
ESRB
ONLINE
BROADBAND ONLY
GS
DINOTOPIA
THE SUNSTONE ODYSSEY
GS
OBOTECH
ATTLECRY
TDK


Who Are You?
ANTHEM™

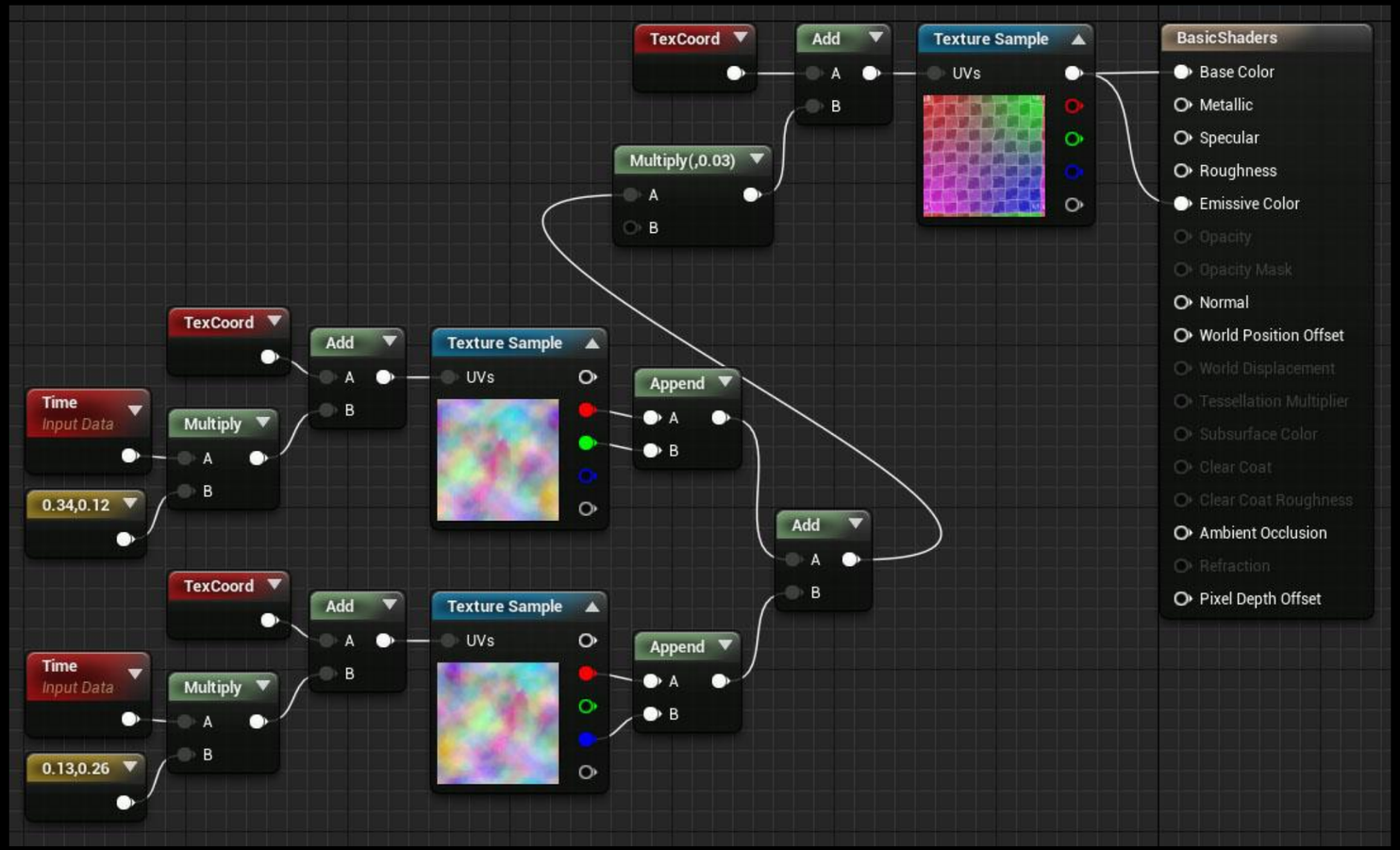
Our Agenda

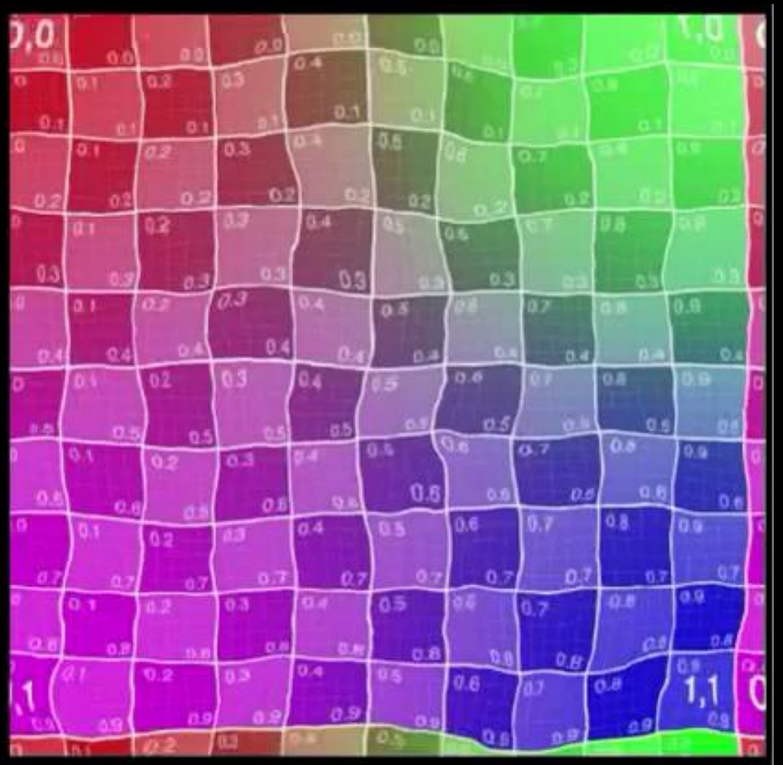
- Review
- Rustling Leaves
- Procedural Noise
  - Snowy Trees Example
  - Rock Layers Example

These concepts can be used in shader code and in node-based editors

Let’s Review!

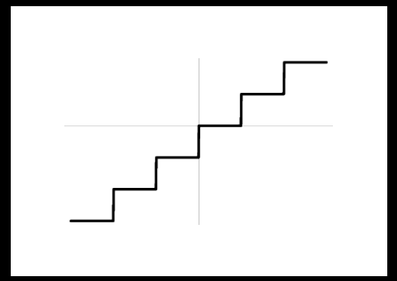
Distortion

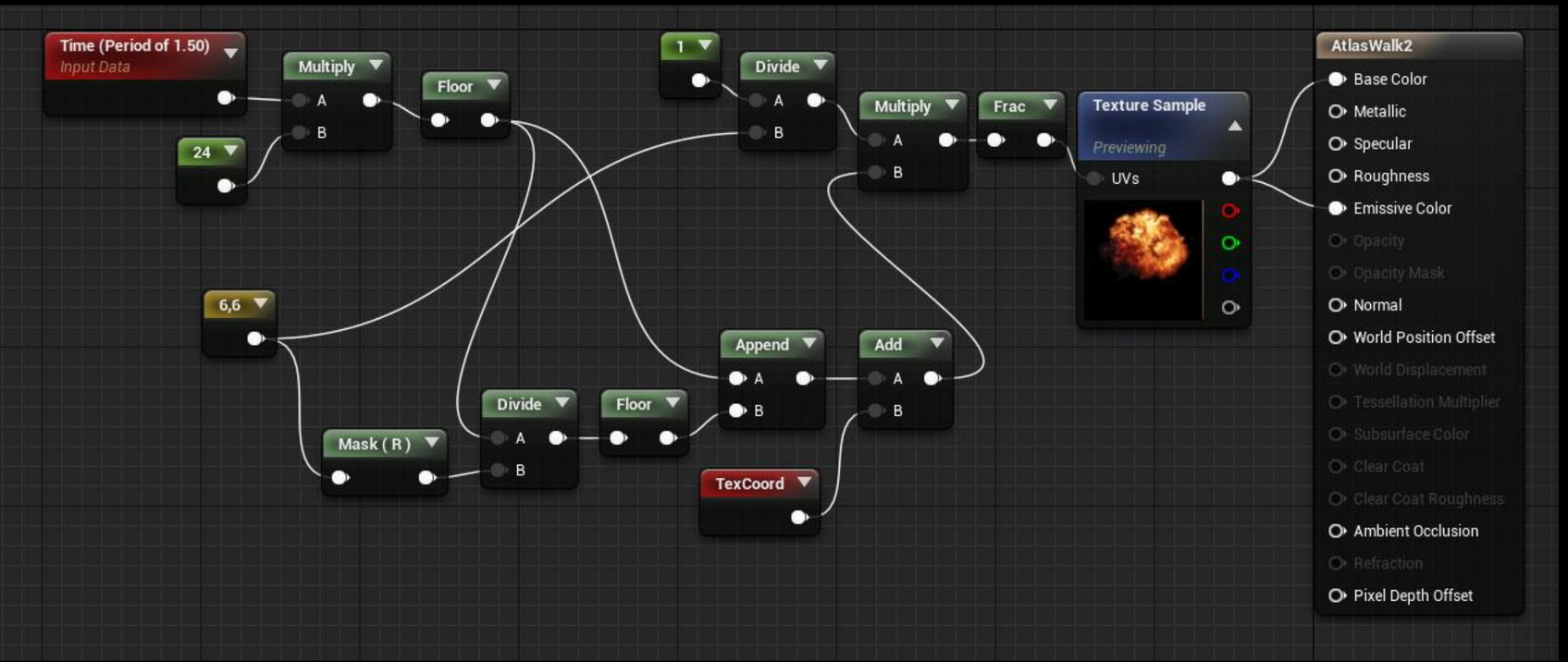

TexCoord
Add
A
B
Texture Sample
UVs
Multiply(,0.03)
A
B
BasicShaders
Base Color
Metallic
Specular
Roughness
Emissive Color
Opacity
Opacity Mask
Normal
World Position Offset
World Displacement
Tessellation Multiplier
Subsurface Color
Clear Coat
Clear Coat Roughness
Ambient Occlusion
Refraction
Pixel Depth Offset
TexCoord
Add
Texture Sample
UVs
Time
Input Data
Multiply
A
B
0.34,0.12
Append
A
B
Add
A
B
TexCoord
Add
Texture Sample
UVs
Time
Input Data
Multiply
A
B
0.13,0.26
Append
A
B



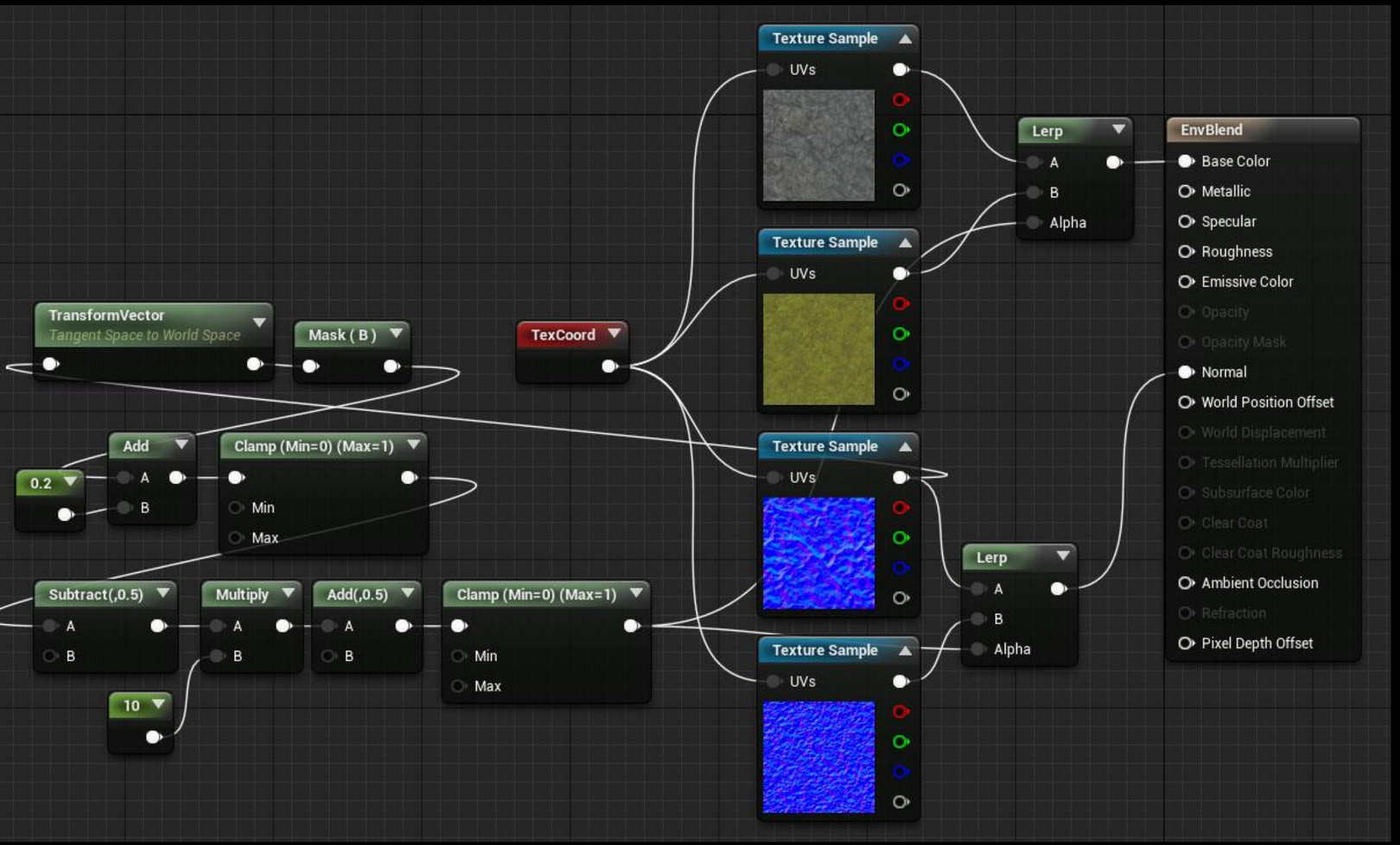
Atlas Walk

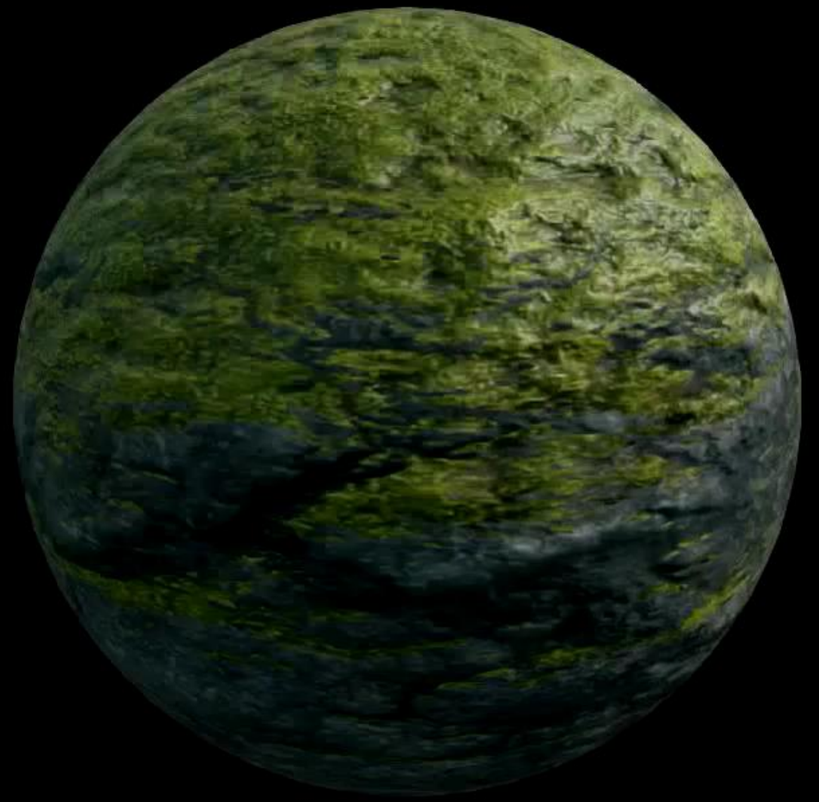

Floor



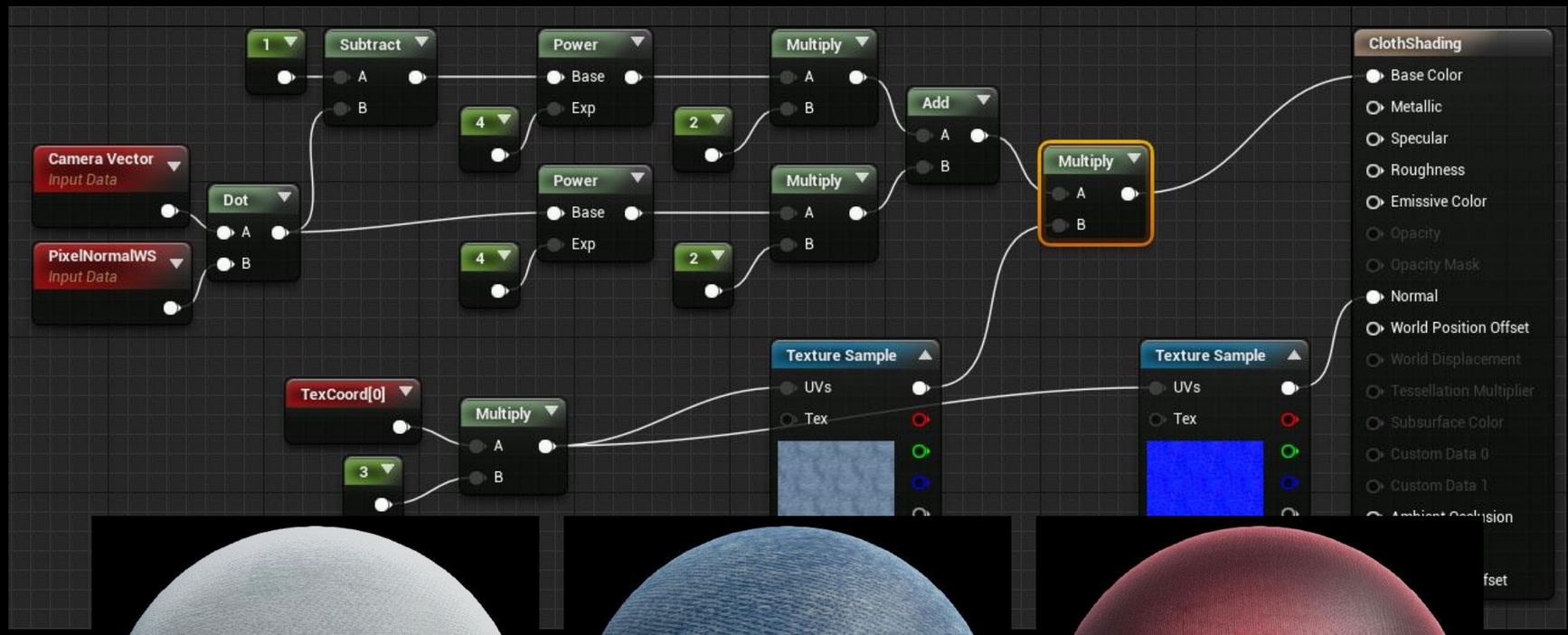

Time (Period of 1.50)
Input Data
Multiply
A
B
Floor
24
1
Divide
A
B
Multiply
A
B
Frac
Texture Sample
Previewing
UVs
6,6
Append
A
B
Add
A
B
Divide
A
B
Floor
Mask (R)
TexCoord
AtlasWalk2
Base Color
Metallic
Specular
Roughness
Emissive Color
Opacity
Opacity Mask
Normal
World Position Offset
World Displacement
Tessellation Multiplier
Subsurface Color
Clear Coat
Clear Coat Roughness
Ambient Occlusion
Refraction
Pixel Depth Offset

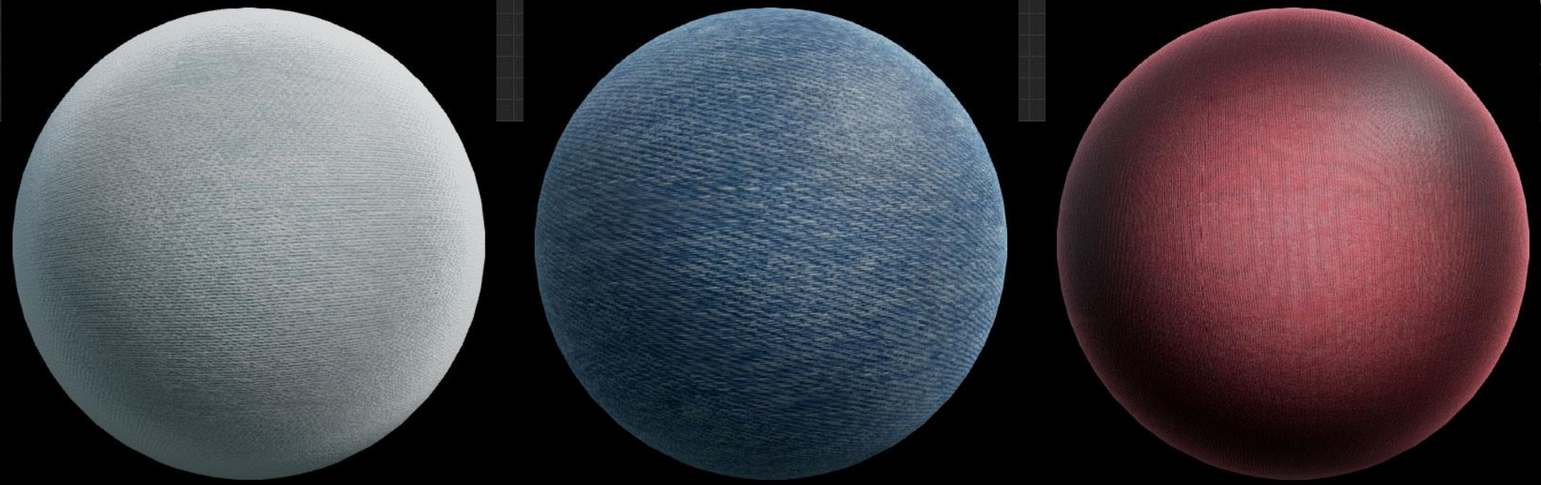
Projection Blend





Simple Cloth



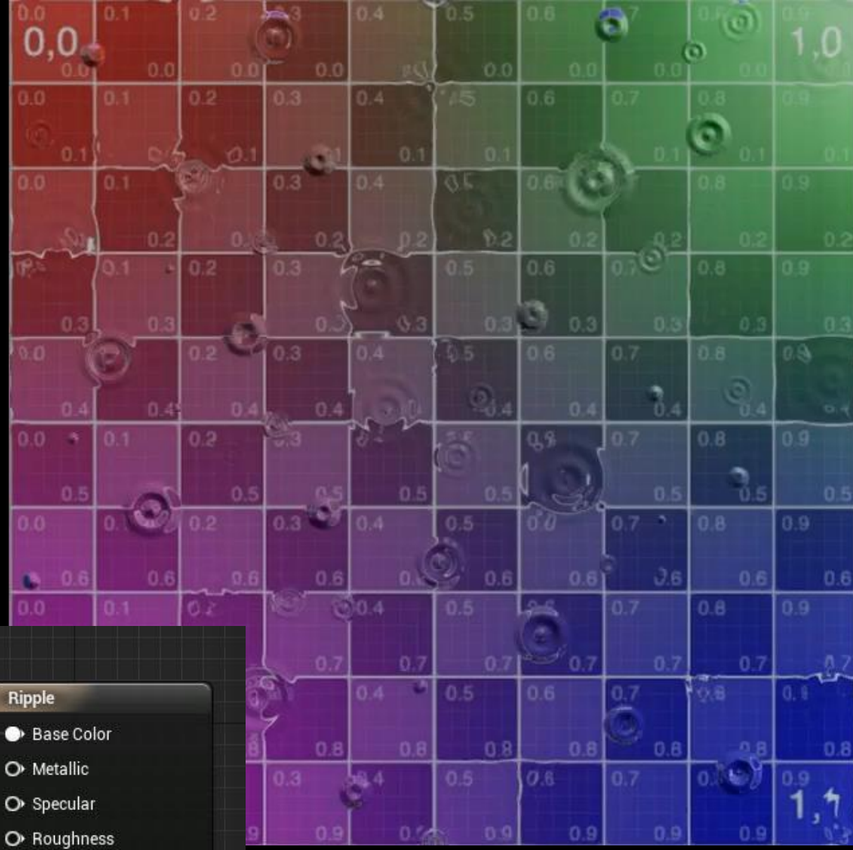


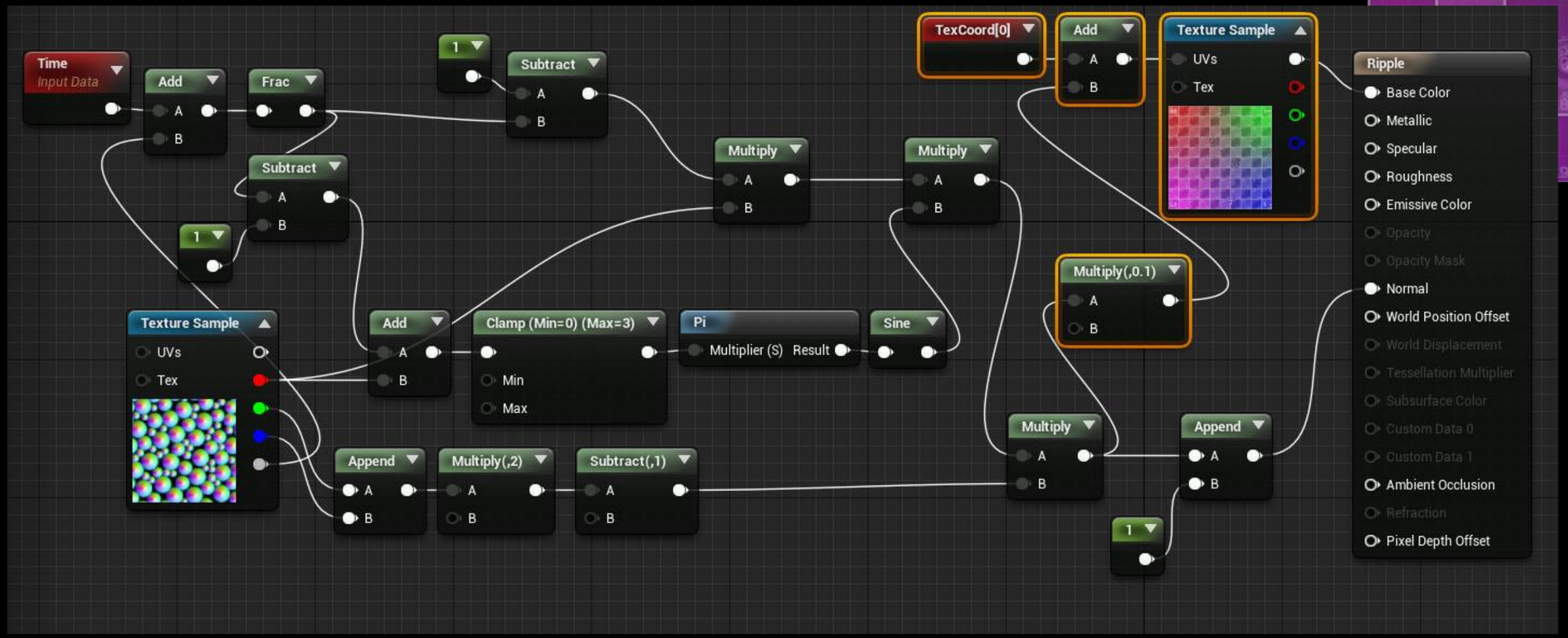
Cotton

Denim

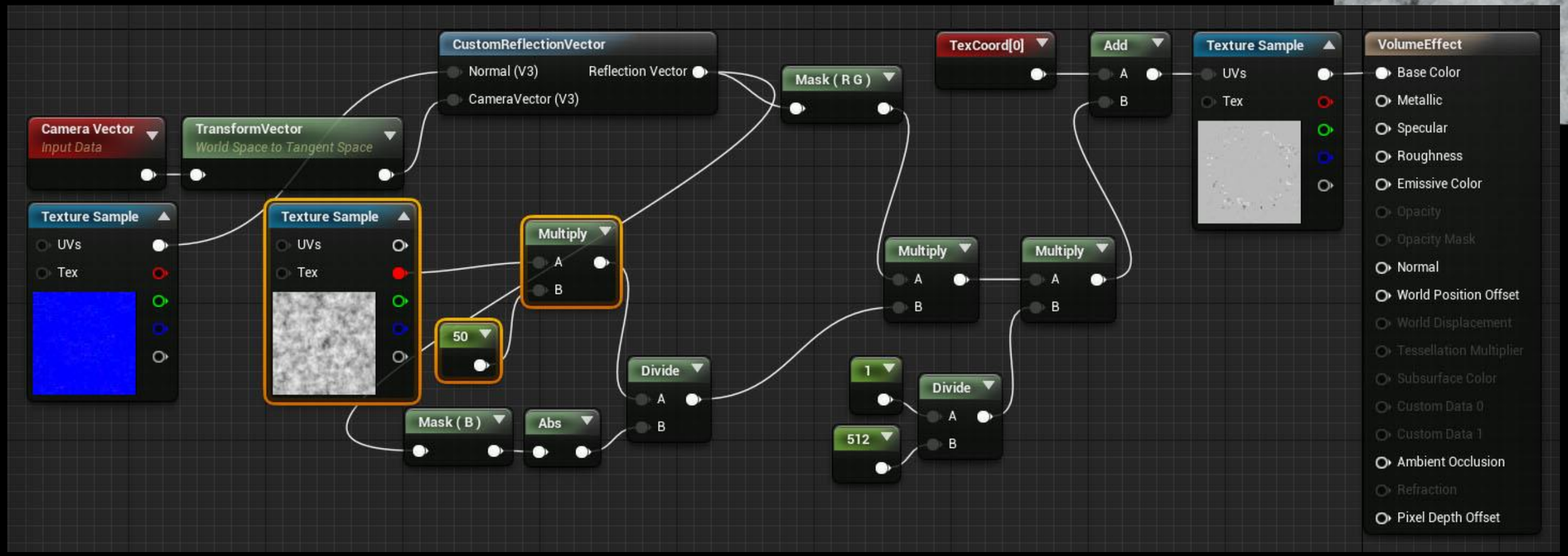
Silk

Rain Ripples





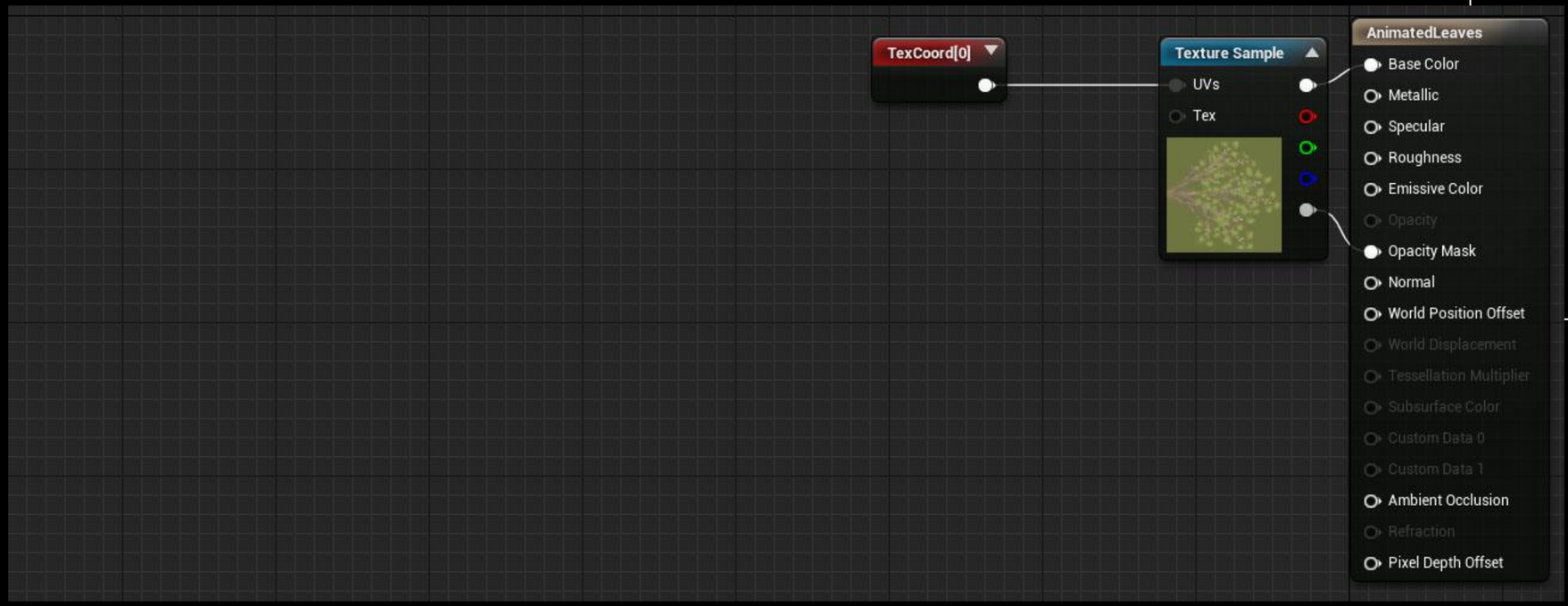
Volumetric Ice



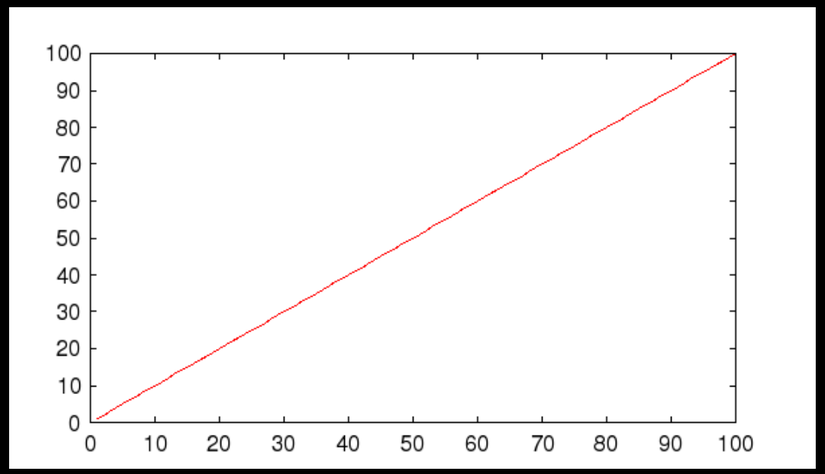
Rustling Leaves

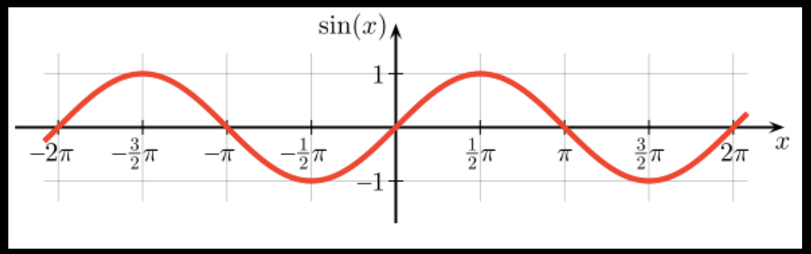
Textures with Multiple Leaves

Simple Shader

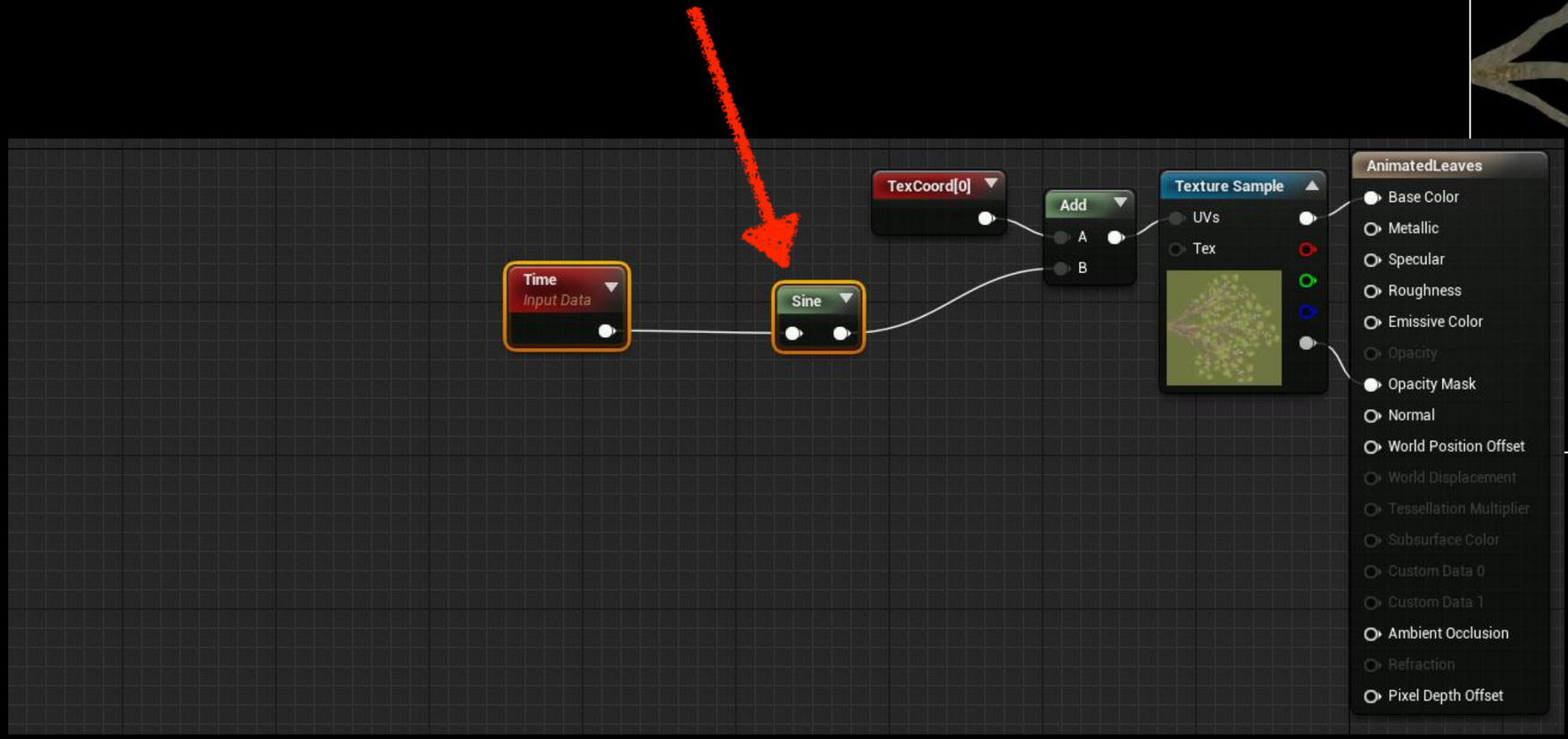


Sine Node Converts Time To Wave

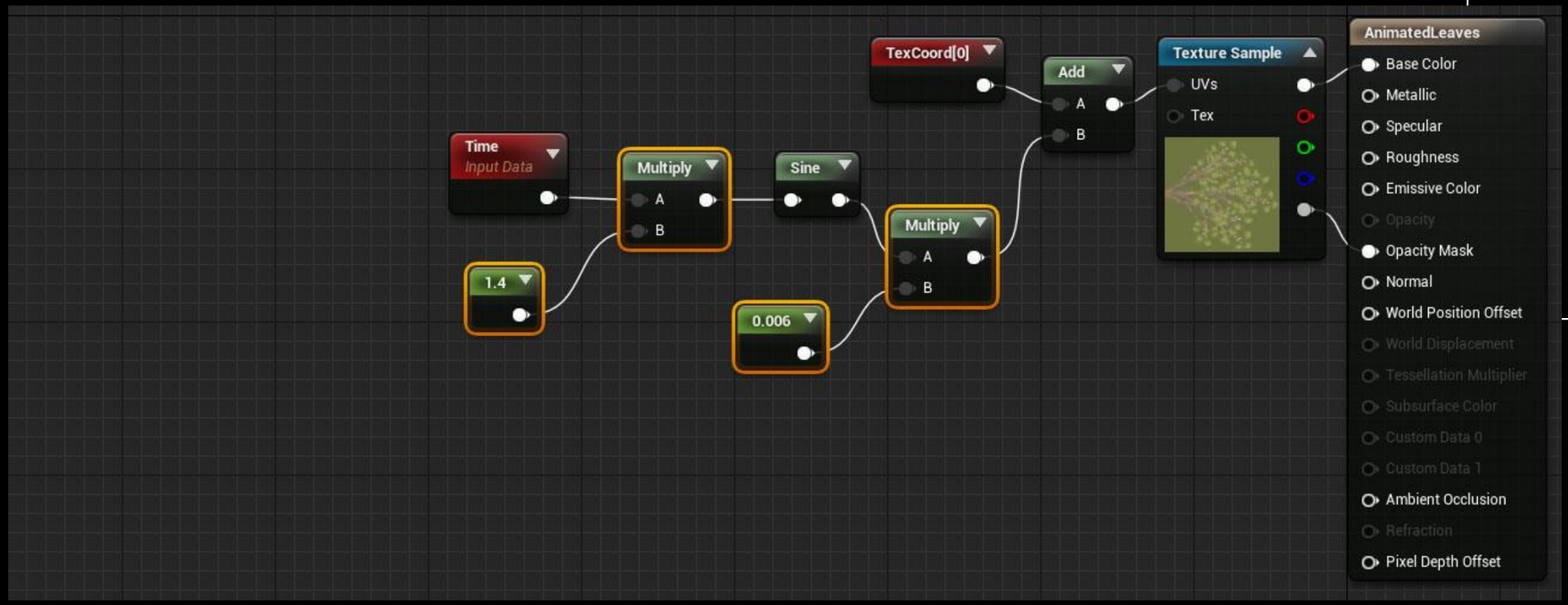




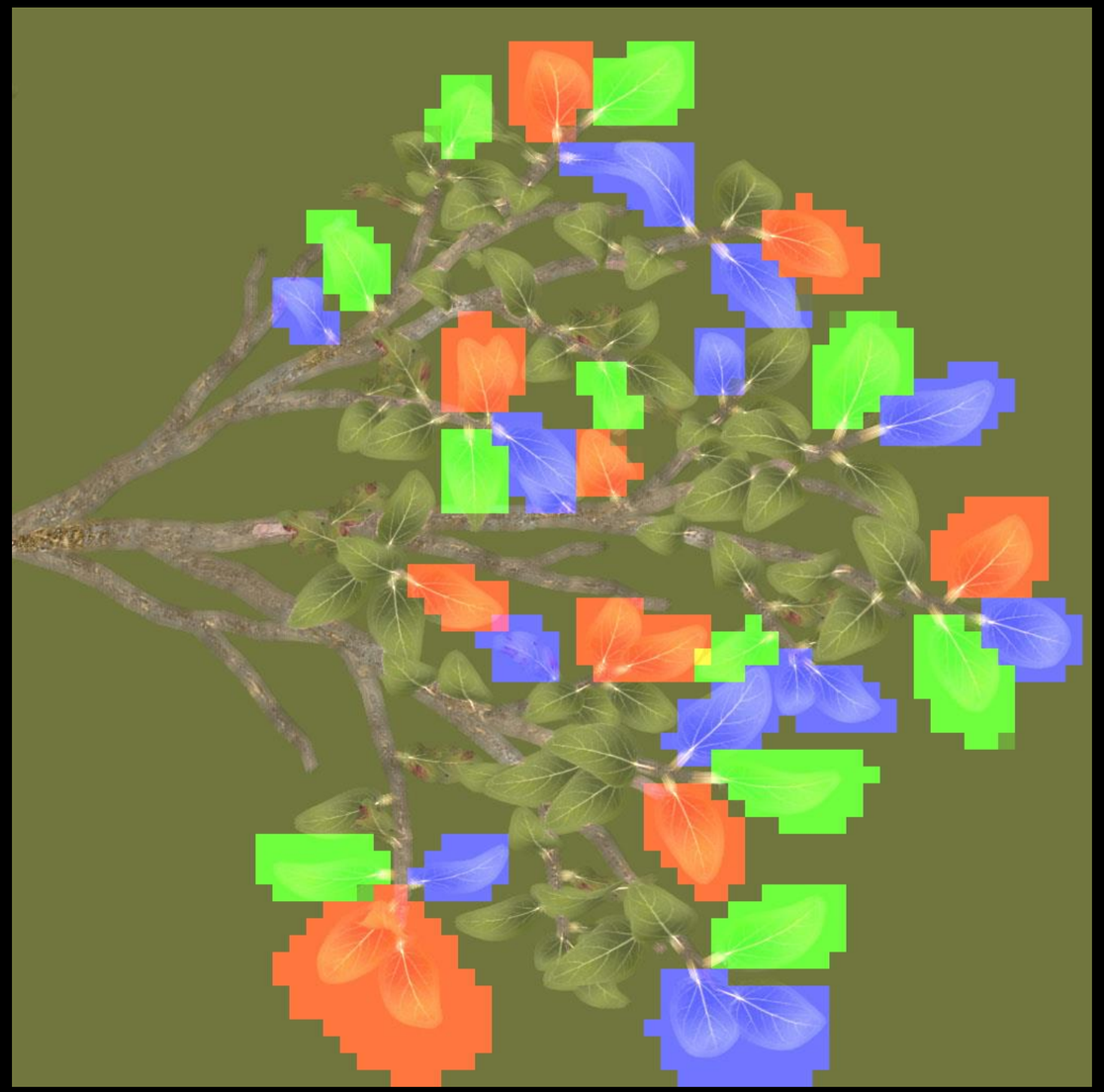
Sine Wave Animation

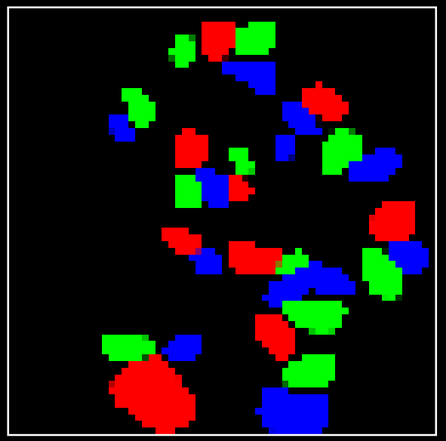


Control Speed and Distance

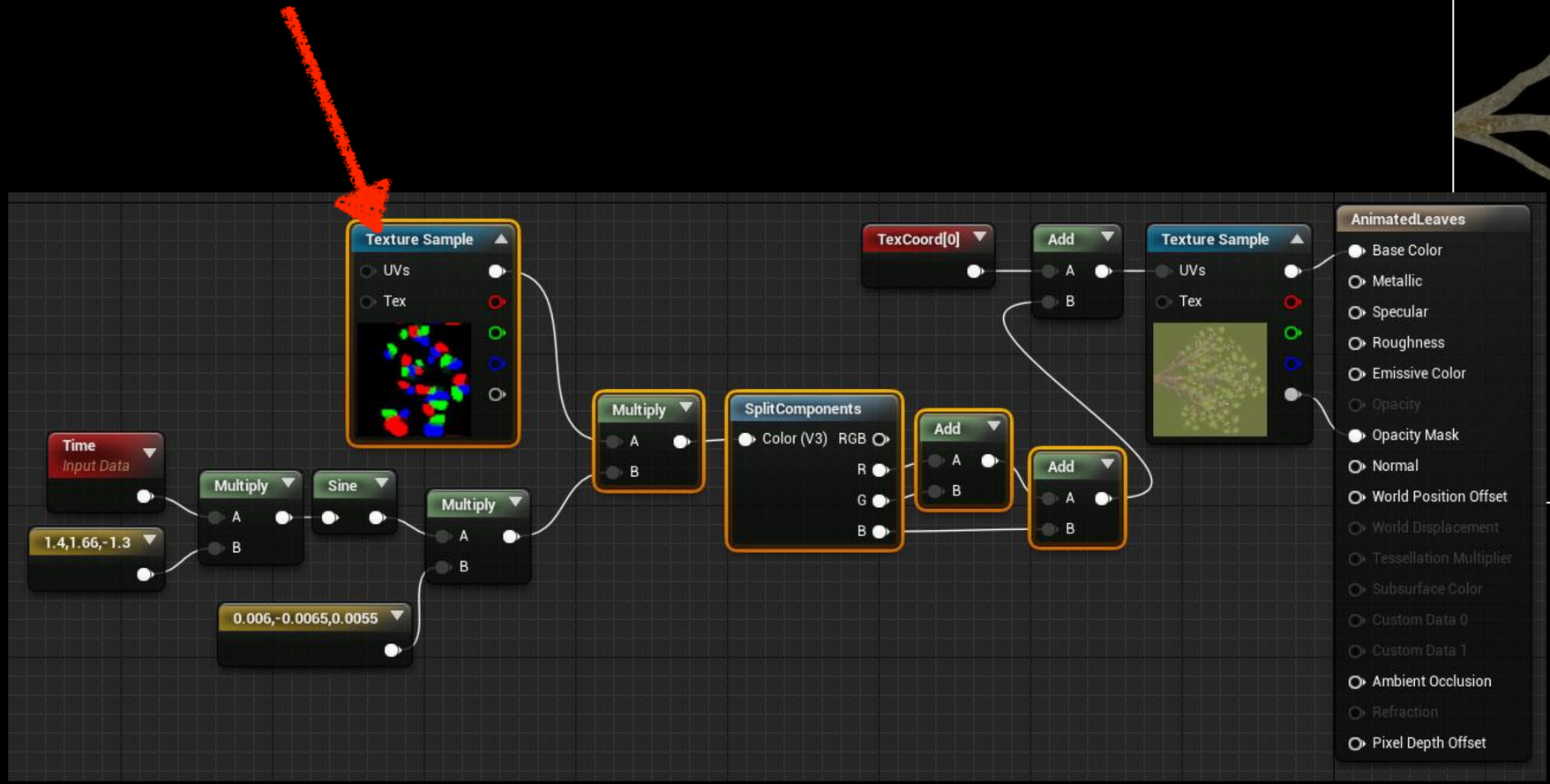


Mask Texture

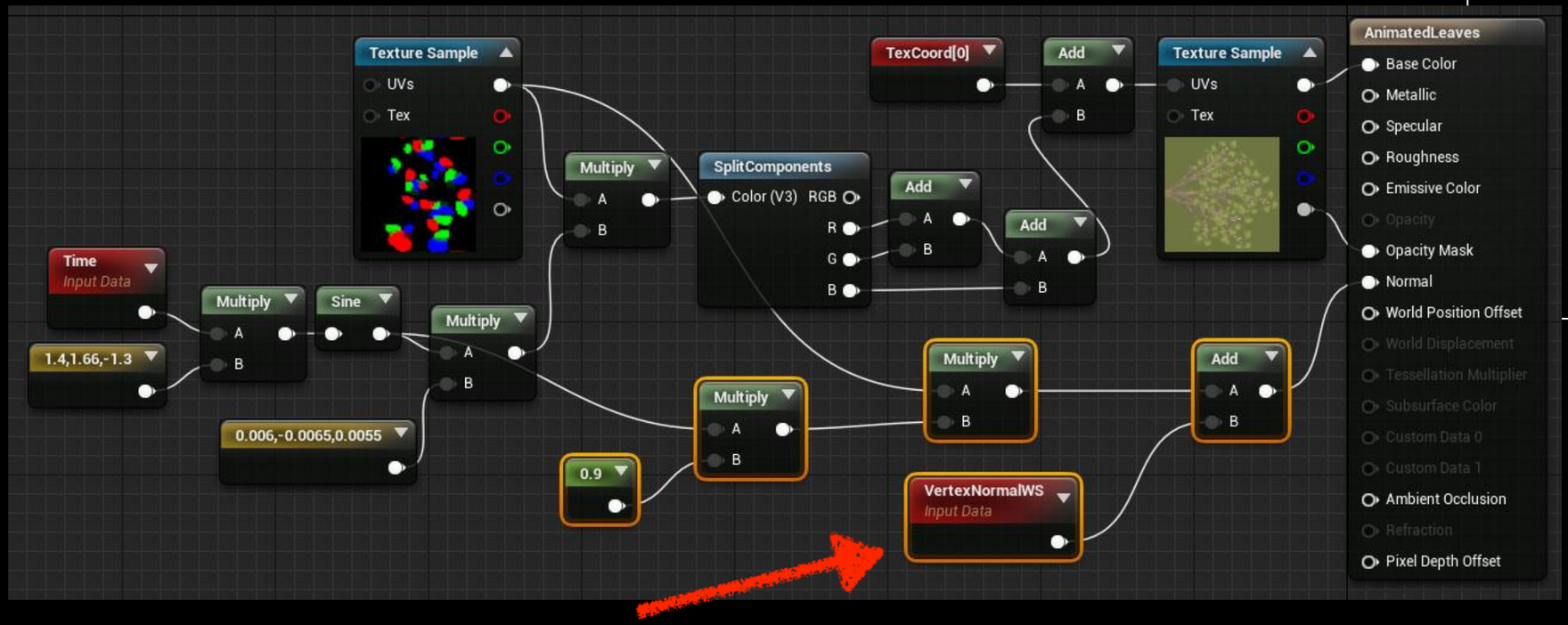




Mask for Unique Movement



Normal Offset



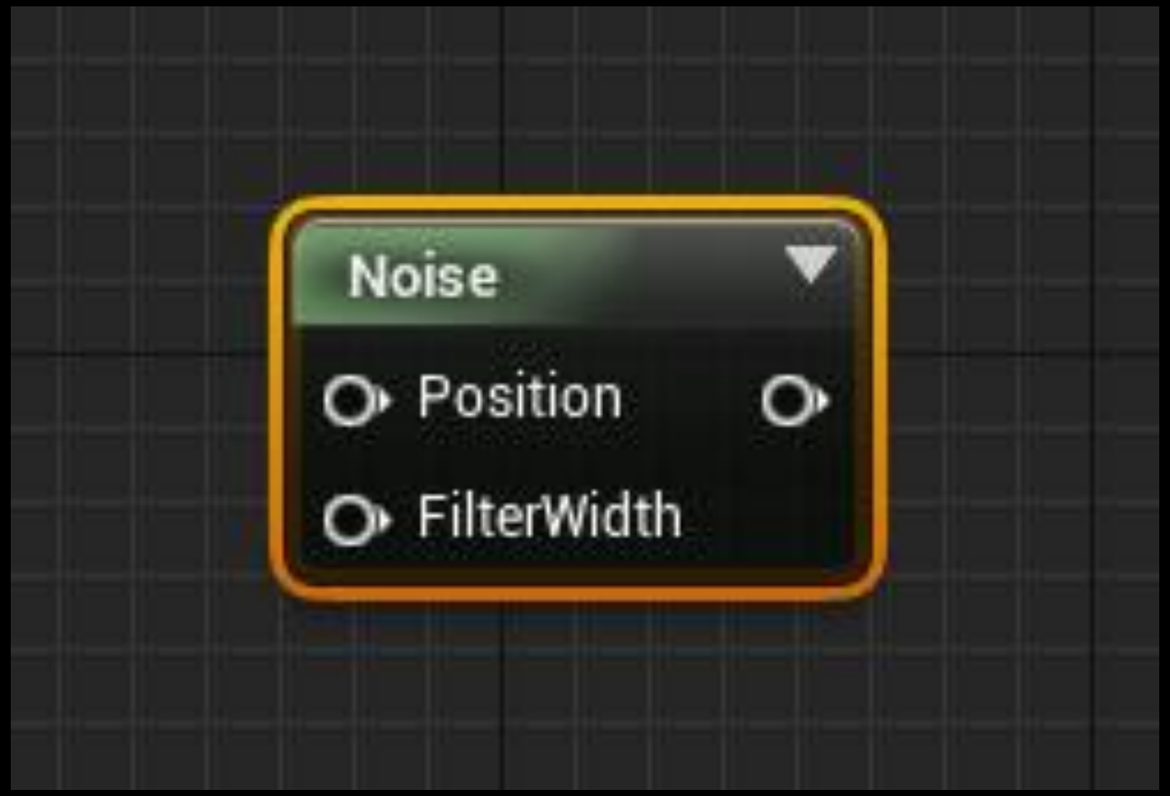

Before


After

Leaf Wiggle Code

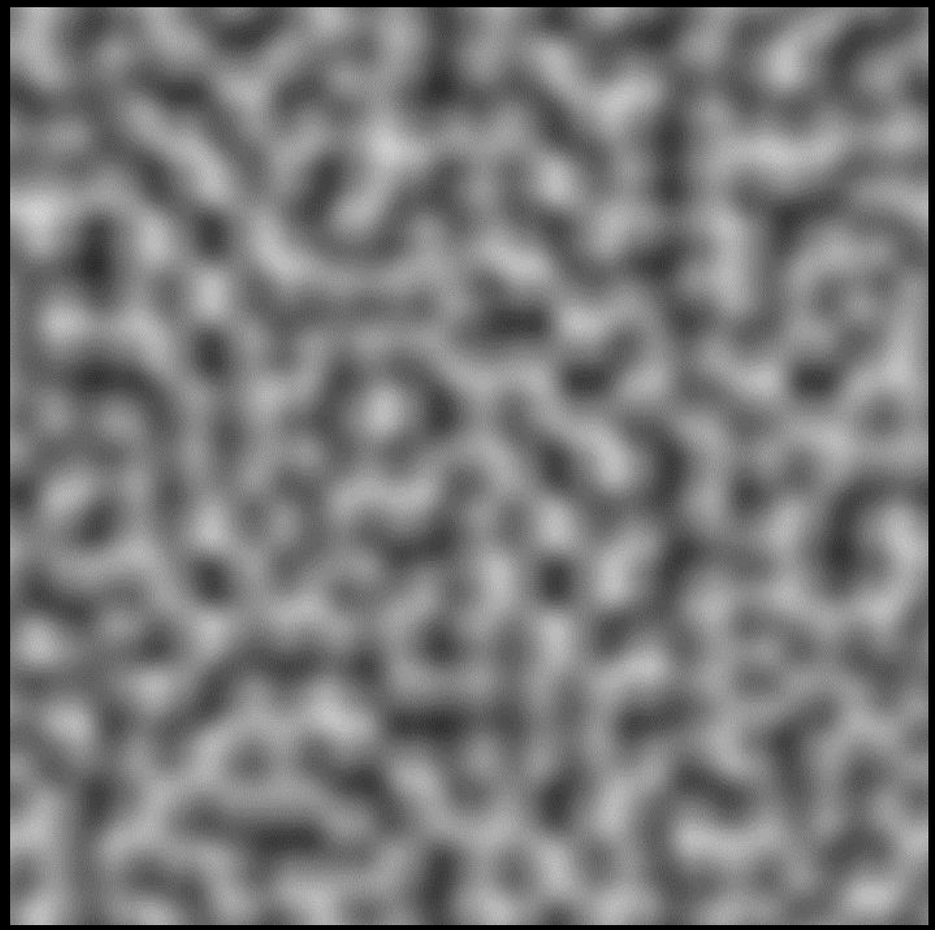
```
float LeafWiggleOffset(float2 UV, float CurrentTime)
{
    float4 wiggleMask = tex2D(wiggleMaskTexture, UV);
    float3 speeds = float3(1.4, 1.66. -1.3) * CurrentTime;
    float3 wiggles = sin(speeds) * float3(0.006, -0.0065, 0.0055) * wiggleMask.rgb;
    return wiggles.r + wiggles.g + wiggles.b;
}
```

Procedural Noise



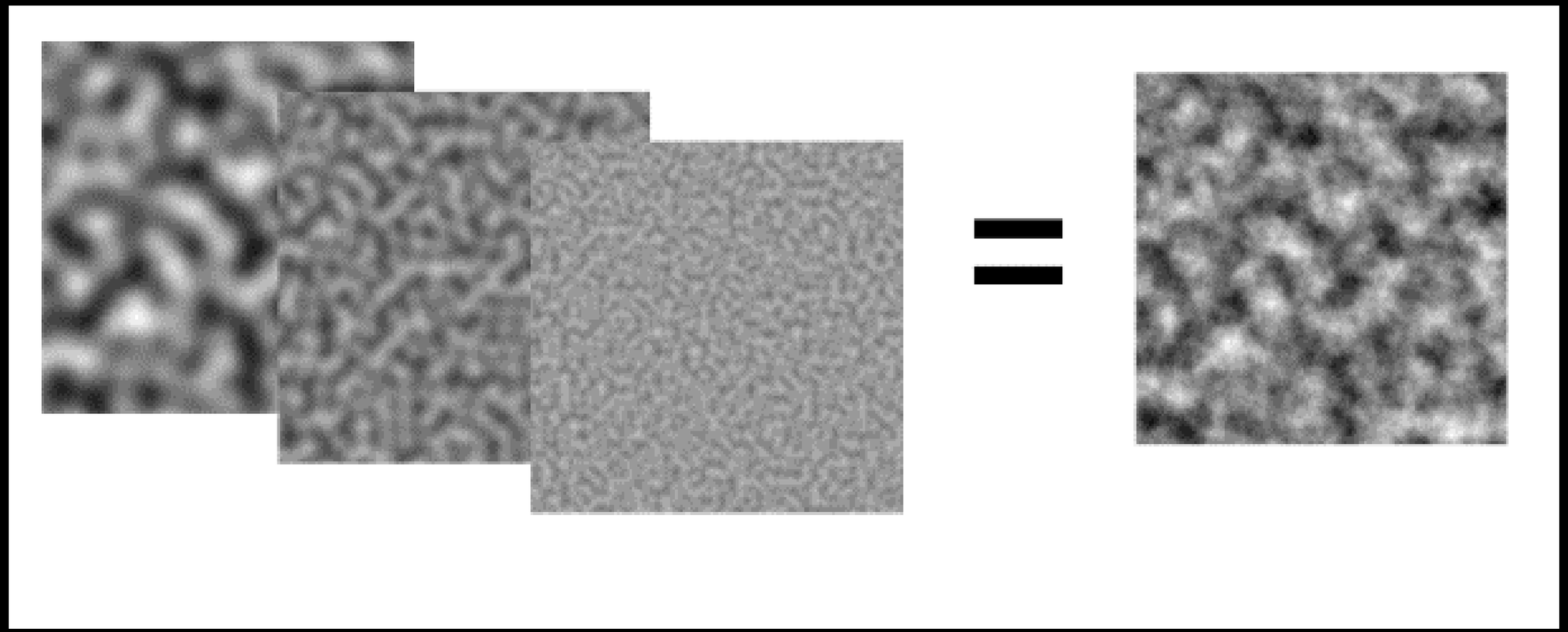
What is it?

- Gradient noise algorithm
- Non-repeating
- Infinite without tiling
- No sharp edges
- Appears random

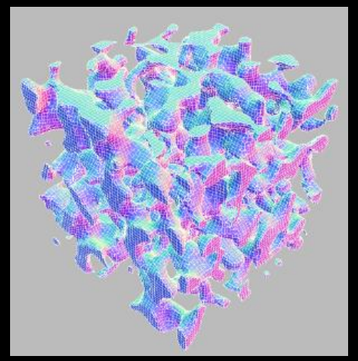
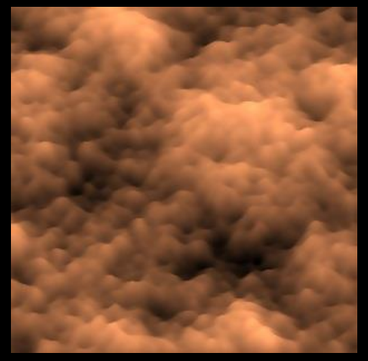


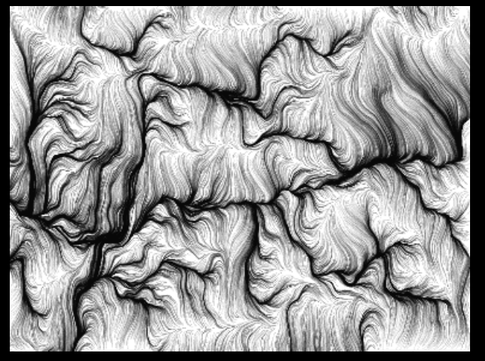
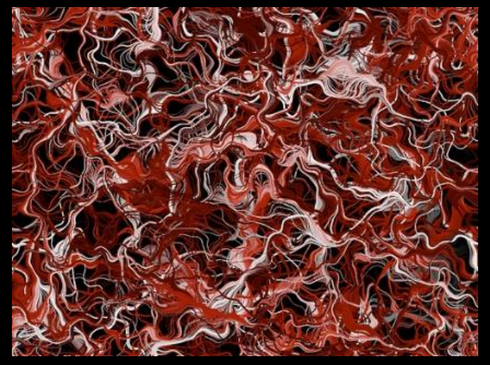
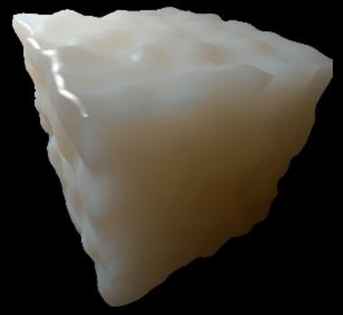
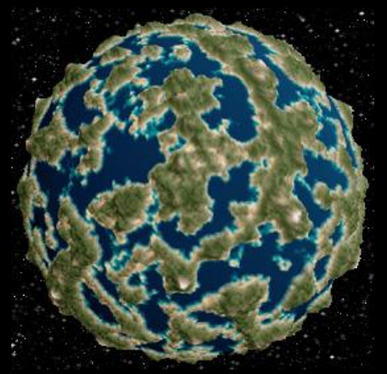
What is it?

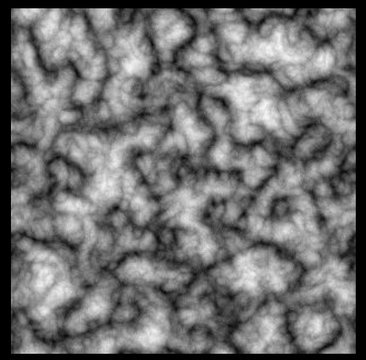
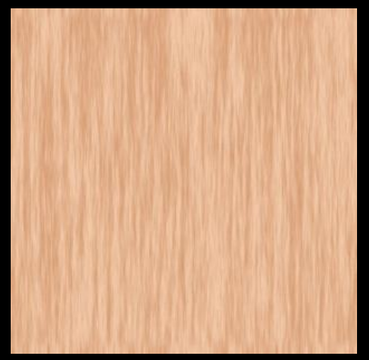
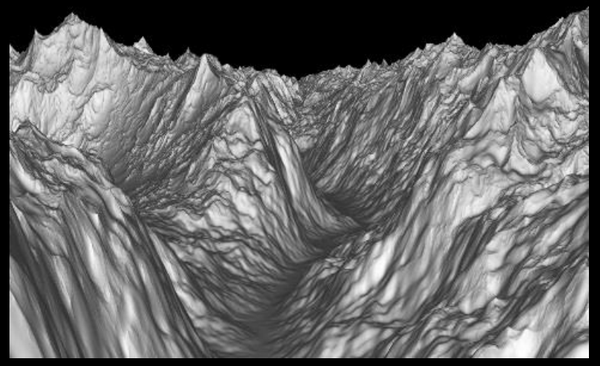
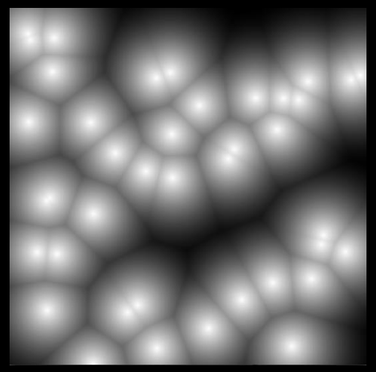
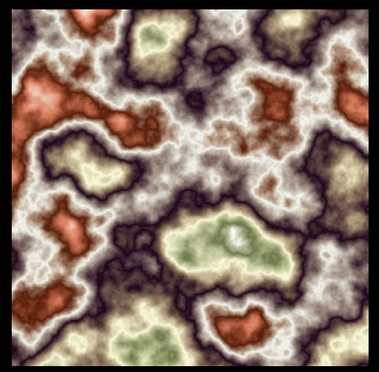
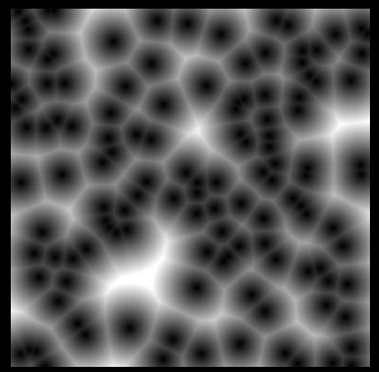
- Used to create fractal patterns

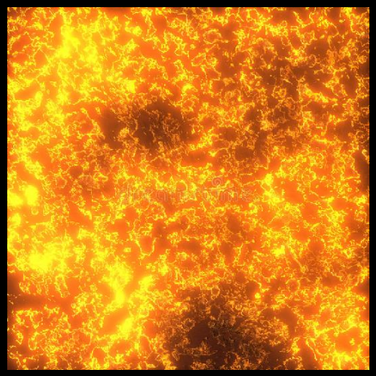


What is it?



Optimized for Real-Time

```
// optimized version
float inoise(float3 p)
{
    float3 P = fmod(floor(p), 256.0);   // FIND UNIT CUBE THAT CONTAINS POINT
    p -= floor(p);                      // FIND RELATIVE X,Y,Z OF POINT IN CUBE.
    float3 f = fade(p);                 // COMPUTE FADE CURVES FOR EACH OF X,Y,Z.

    P = P / 256.0;
    const float one = 1.0 / 256.0;

    // HASH COORDINATES OF THE 8 CUBE CORNERS
    float4 AA = perm2d(P.xy) + P.z;

    // AND ADD BLENDED RESULTS FROM 8 CORNERS OF CUBE
    return lerp( lerp( lerp( gradperm(AA.x, p ),
                             gradperm(AA.z, p + float3(-1, 0, 0) ), f.x),
                       lerp( gradperm(AA.y, p + float3(0, -1, 0) ),
                             gradperm(AA.w, p + float3(-1, -1, 0) ), f.x), f.y),

                 lerp( lerp( gradperm(AA.x+one, p + float3(0, 0, -1) ),
                             gradperm(AA.z+one, p + float3(-1, 0, -1) ), f.x),
                       lerp( gradperm(AA.y+one, p + float3(0, -1, -1) ),
                             gradperm(AA.w+one, p + float3(-1, -1, -1) ), f.x), f.y), f.z);
}
```

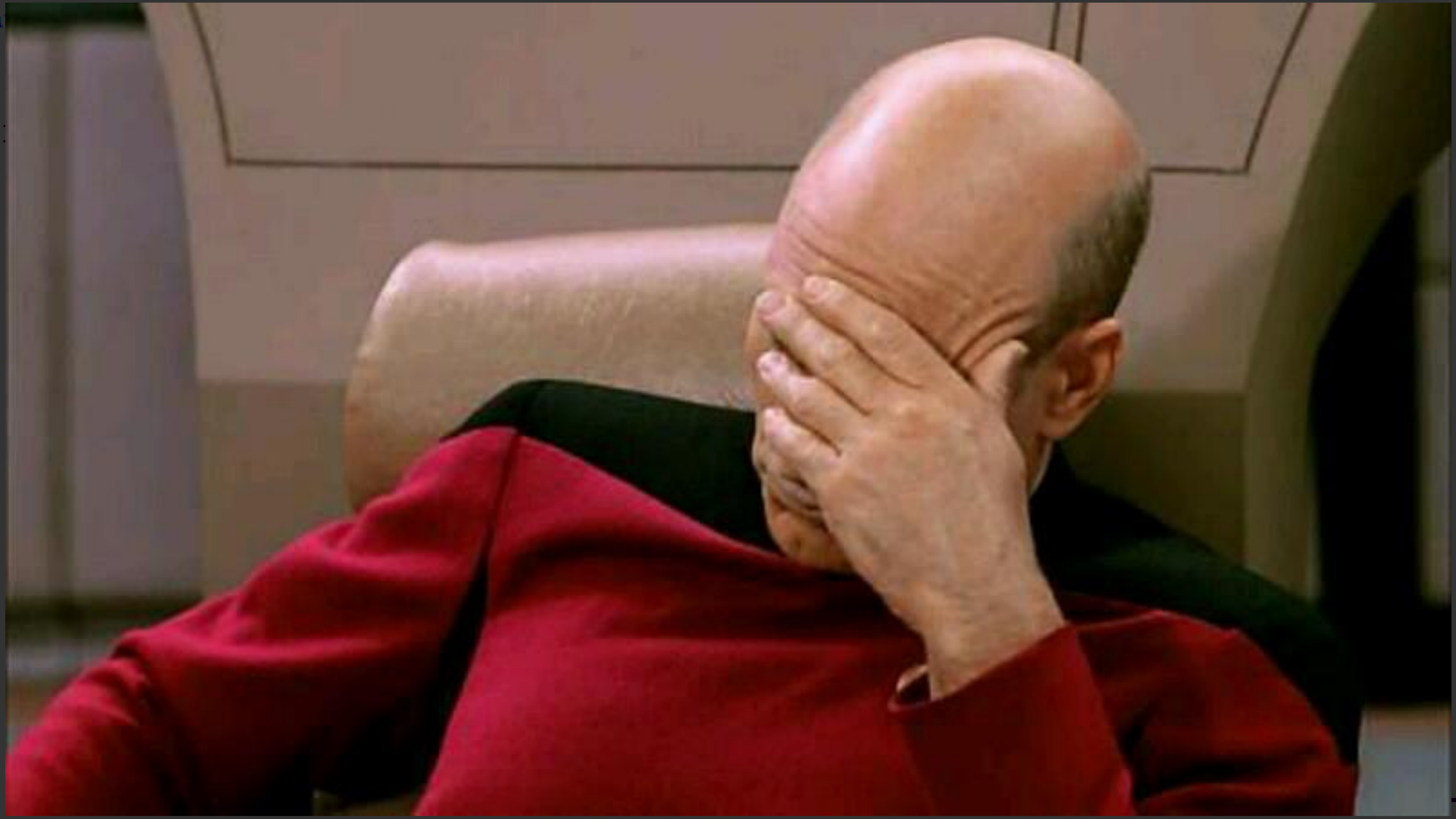
Optimized for Real-Time

```
// optimized version
float inoise(float3 p)
{
    float3 P = fmod(floor(p), 256.0);   // FIND UNIT CUBE THAT CONTAINS POINT
    p -= floor(p);                      // FIND RELATIVE X,Y,Z OF POINT IN CUBE.
    float3 f = fade(p);                 // COMPUTE FADE CURVES FOR EACH OF X,Y,Z.
    P = P / 256.0;
    const float one = 1.0 / 256.0;

    // HASH COORDINATES OF THE 8 CUBE CORNERS
    float4 AA = perm(P.xy) + P.z;

    // AND ADD BLENDED RESULTS FROM 8 CORNERS OF CUBE
    return lerp( lerp( lerp( gradperm(AA.x, p ),
                             gradperm(AA.z, p + float3(-1, 0, 0) ), f.x),
                       lerp( gradperm(AA.y, p + float3(0, -1, 0) ),
                             gradperm(AA.w, p + float3(-1, -1, 0) ), f.x), f.y),

                 lerp( lerp( gradperm(AA.x+one, p + float3(0, 0, -1) ),
                             gradperm(AA.z+one, p + float3(-1, 0, -1) ), f.x),
                       lerp( gradperm(AA.y+one, p + float3(0, -1, -1) ),
                             gradperm(AA.w+one, p + float3(-1, -1, -1) ), f.x), f.y), f.z);
}
```

- **61 Pixel shader instructions**
- **8 Texture Samples**
- **Multiplied by the number of octaves**

Optimized for Real-Time

```
// optimized version
floa
{
```

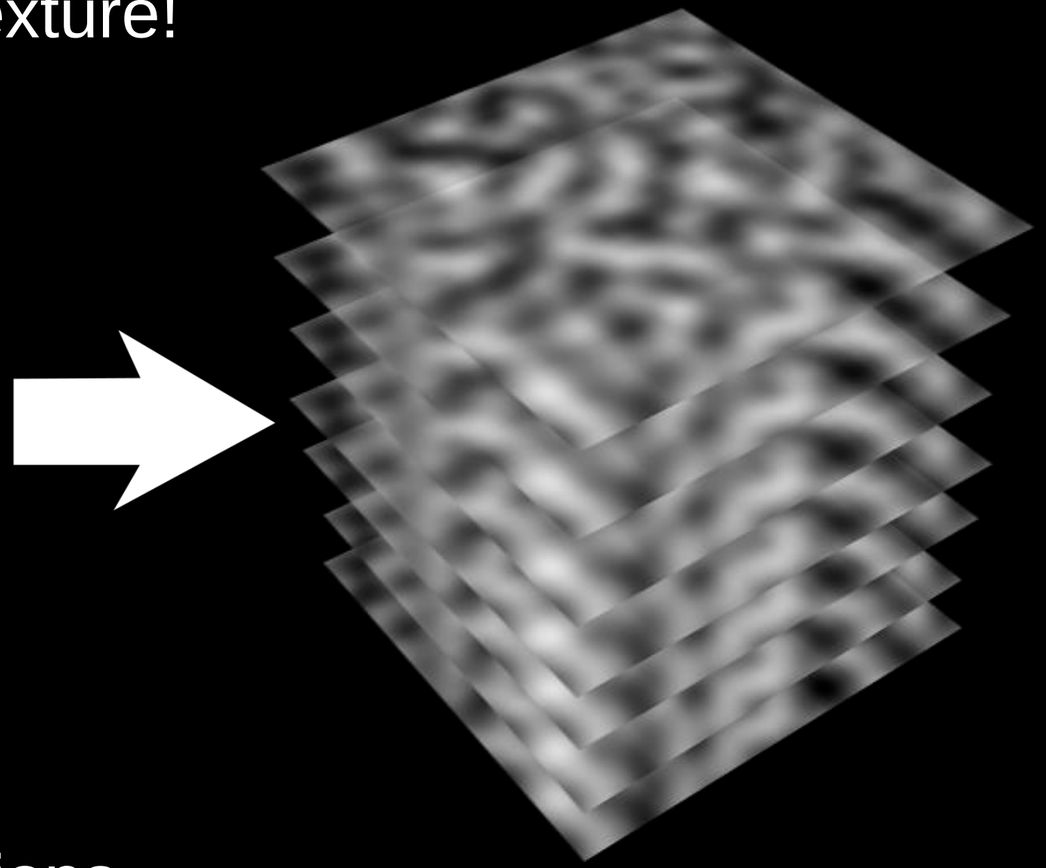


```
.z);
}
```

Optimized for Real-Time

Convert it to a volume texture!

```
// optimized version
float inoise(float3 p)
{
    float3 P = fmod(floor(p), 256.0);   // FIND UNIT CUBE THAT CONTAINS POINT
    p -= floor(p);                      // FIND RELATIVE X,Y,Z OF POINT IN CUBE.
    float3 f = fade(p);                 // COMPUTE FADE CURVES FOR EACH OF X,Y,Z.

    P = P / 256.0;
    const float one = 1.0 / 256.0;

    // HASH COORDINATES OF THE 8 CUBE CORNERS
    float4 AA = perm2d(P.xy) + P.z;

    // AND ADD BLENDED RESULTS FROM 8 CORNERS OF CUBE
    return lerp( lerp( lerp( gradperm(AA.x, p ),
                             gradperm(AA.z, p + float3(-1, 0, 0) ), f.x),
                       lerp( gradperm(AA.y, p + float3(0, -1, 0) ),
                             gradperm(AA.w, p + float3(-1, -1, 0) ), f.x), f.y),

                 lerp( lerp( gradperm(AA.x+one, p + float3(0, 0, -1) ),
                             gradperm(AA.z+one, p + float3(-1, 0, -1) ), f.x),
                       lerp( gradperm(AA.y+one, p + float3(0, -1, -1) ),
                             gradperm(AA.w+one, p + float3(-1, -1, -1) ), f.x), f.y), f.z);
}
```

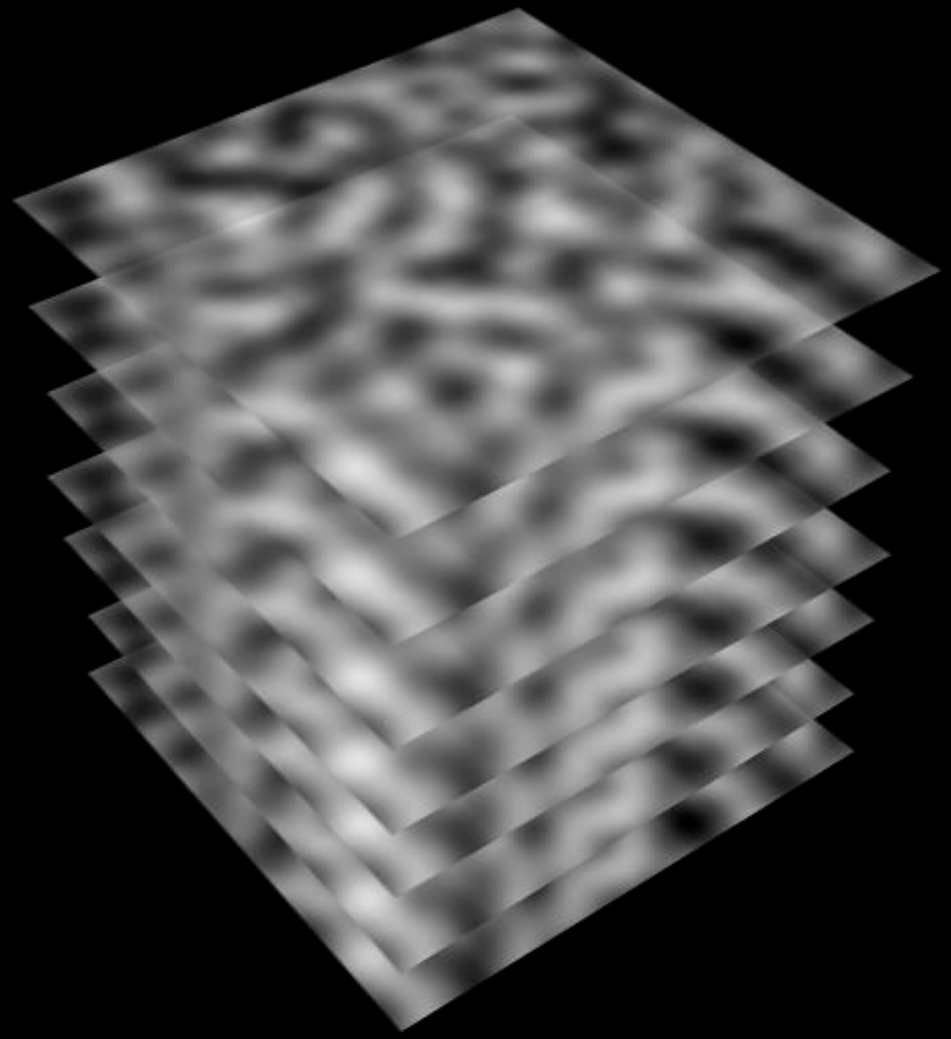


61 Pixel Shader Instructions
8 Texture Samples

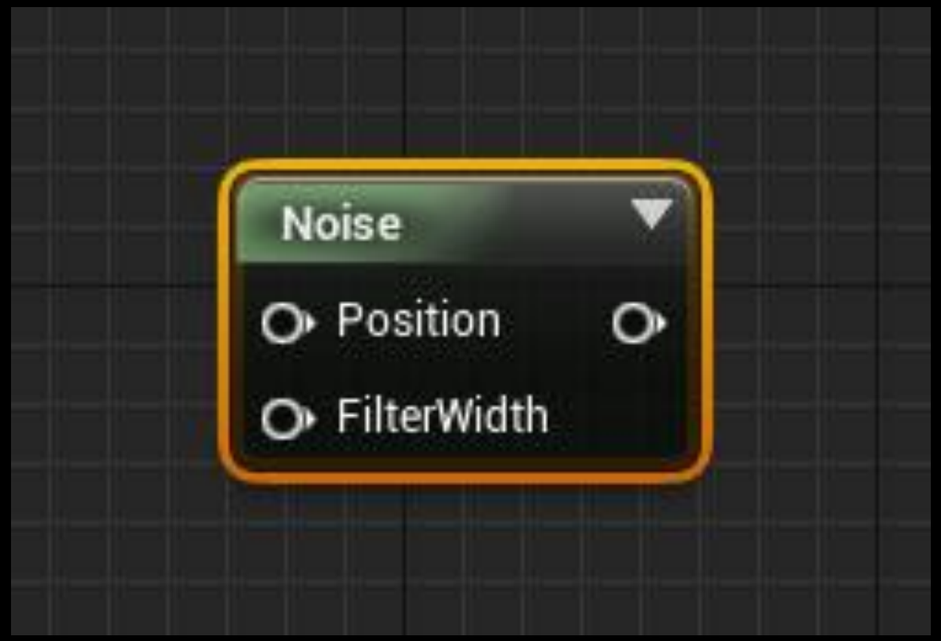
1 Texture Sample

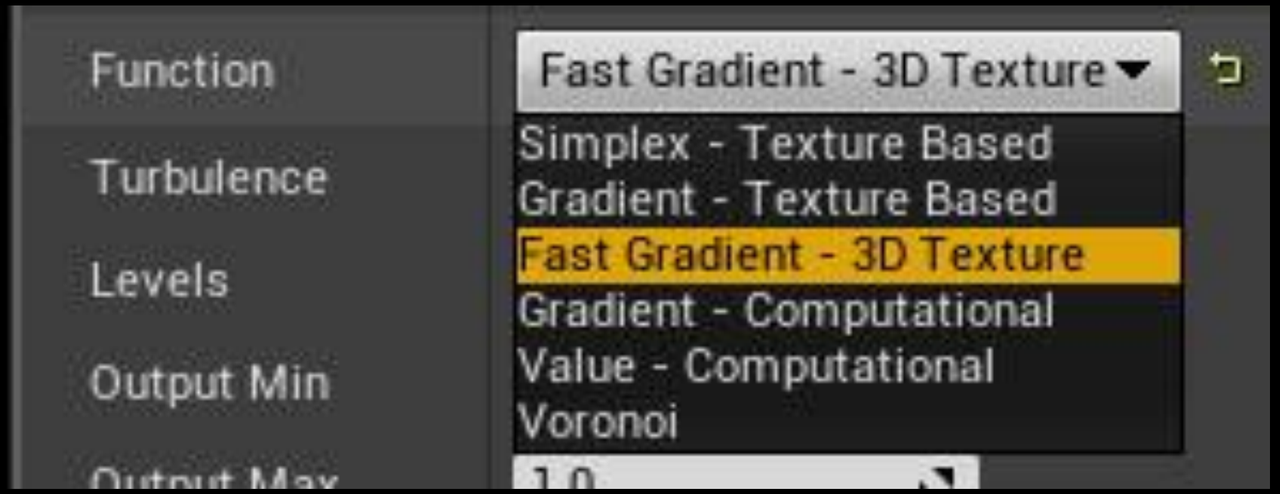
Pros and Cons of Volume Texture

- Pros
  - Faster and Cheaper!
- Cons
  - No longer infinite and random



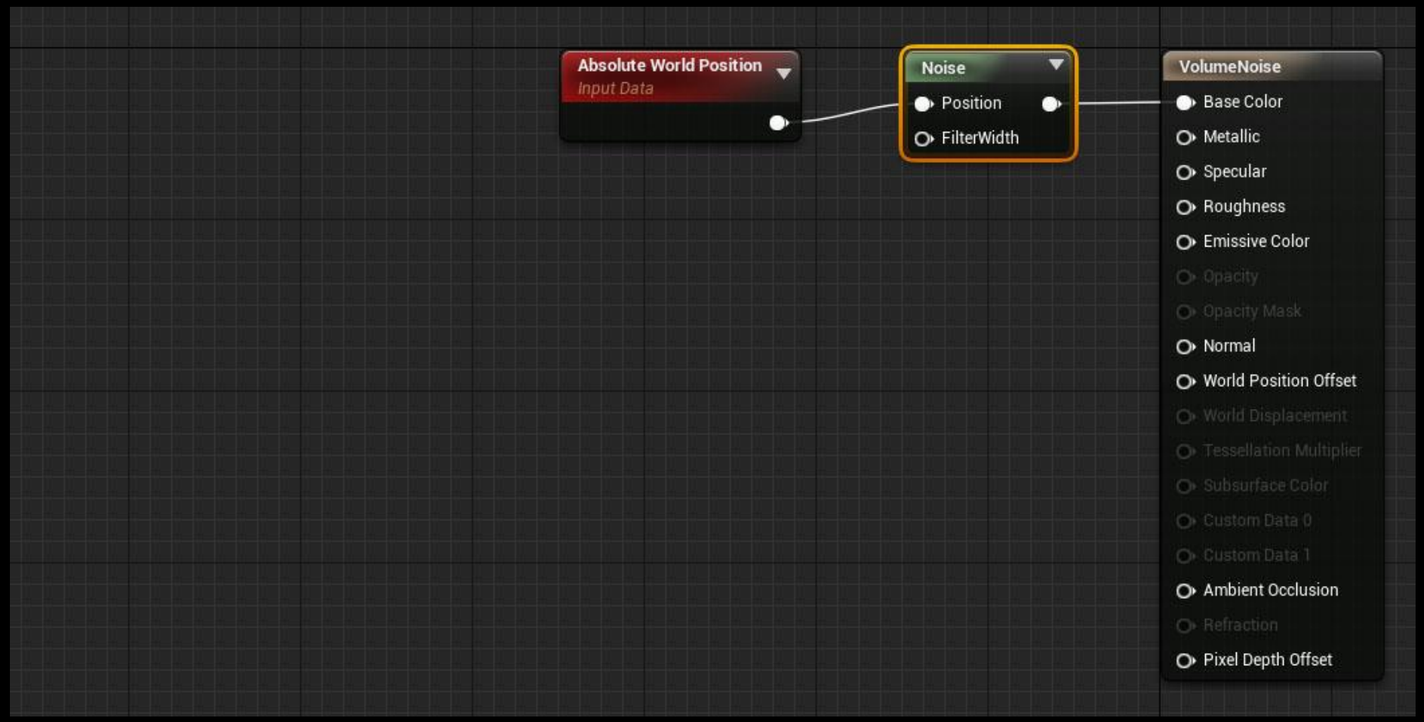
Back to Unreal



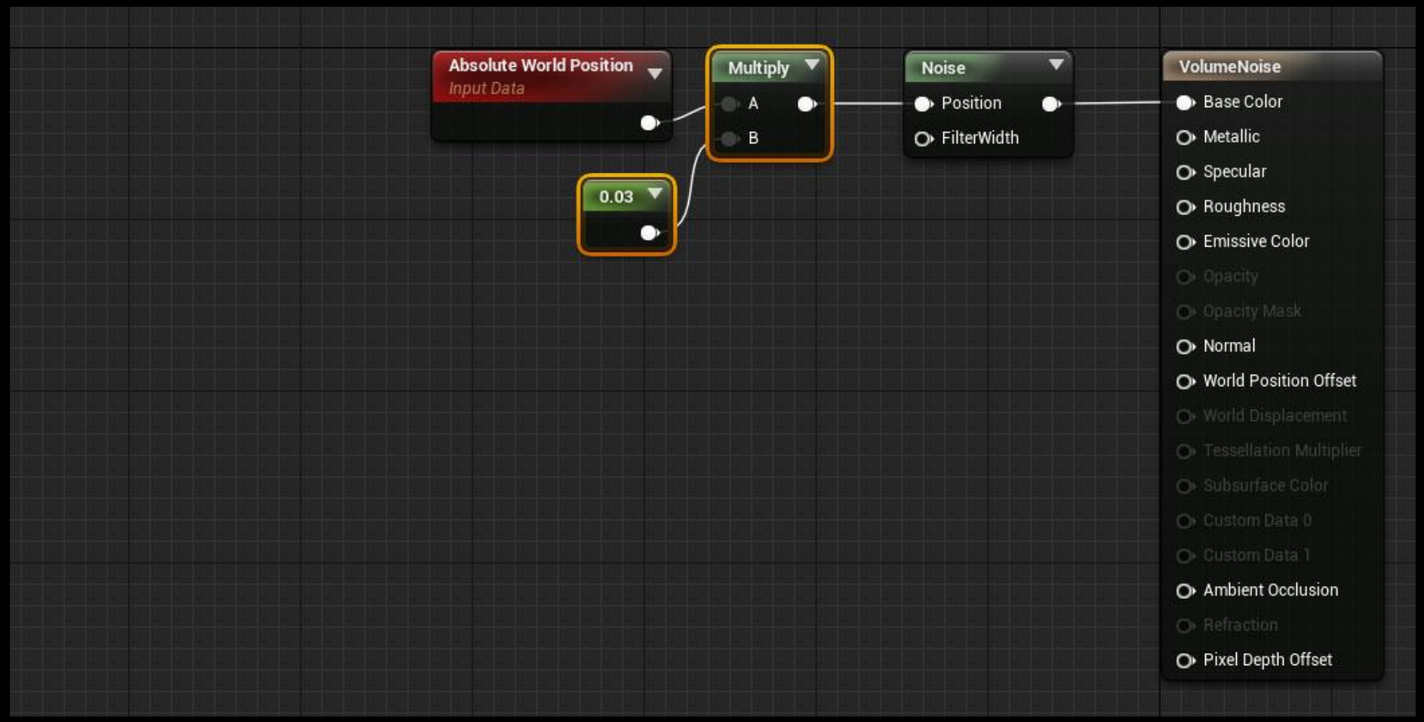


Snowy Trees

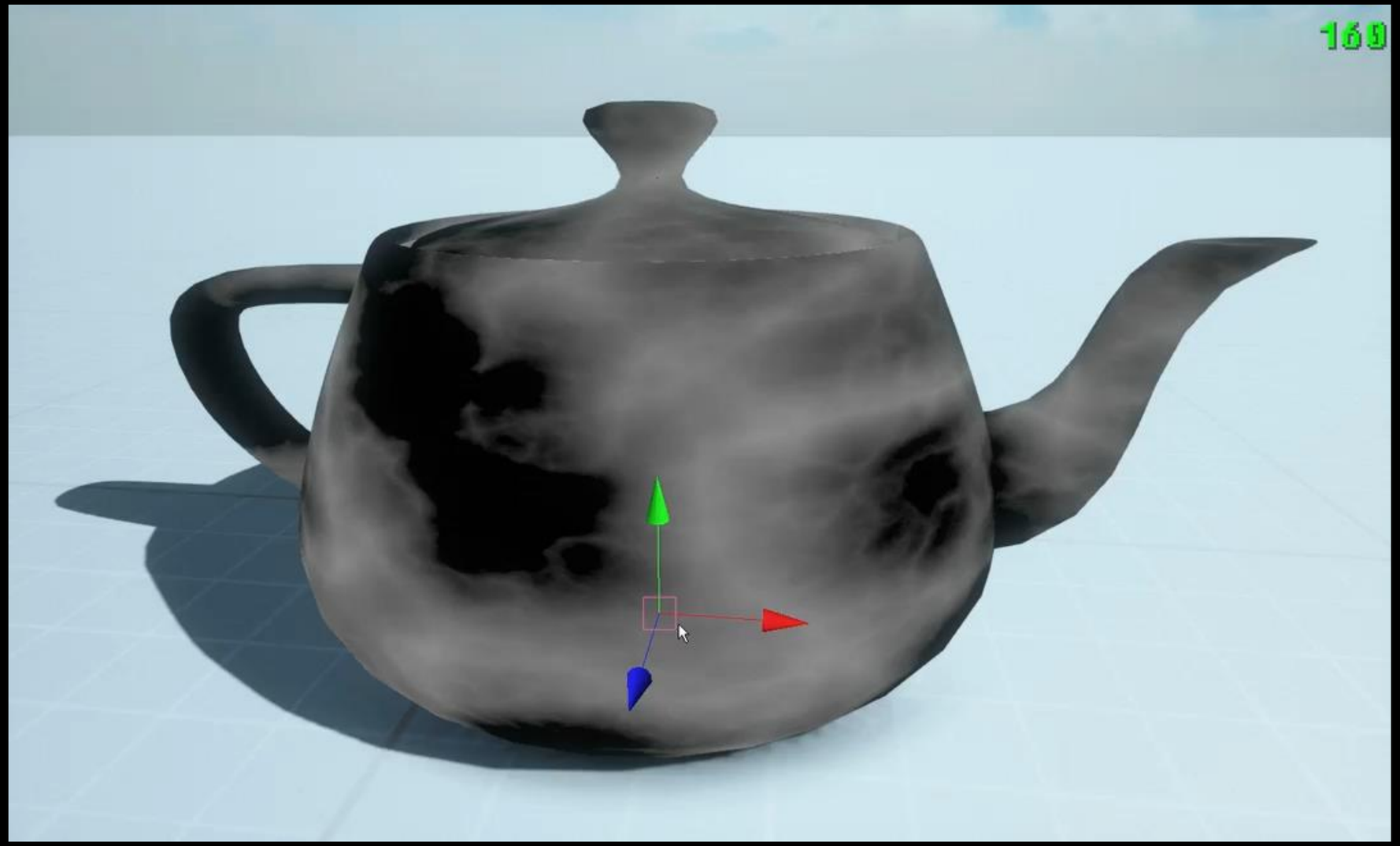
Sample Noise With World Position



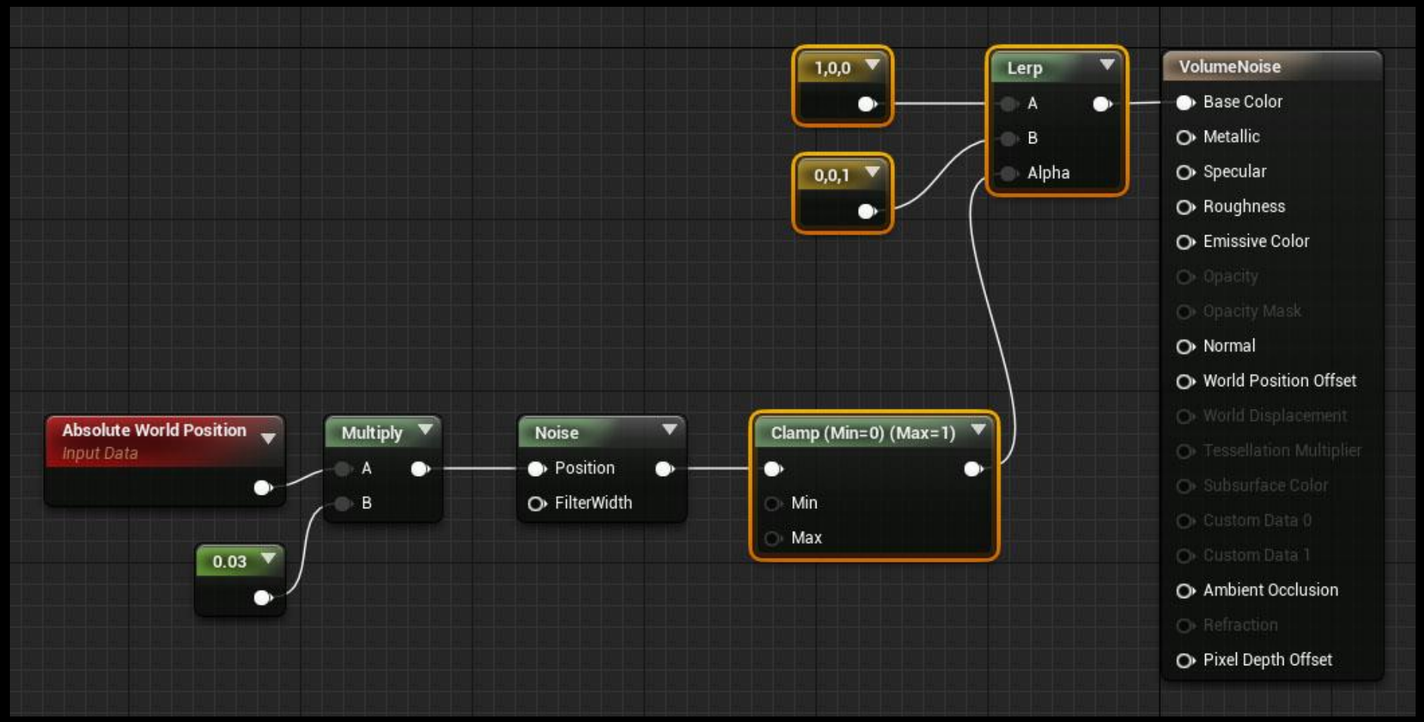
Scale World Position Down



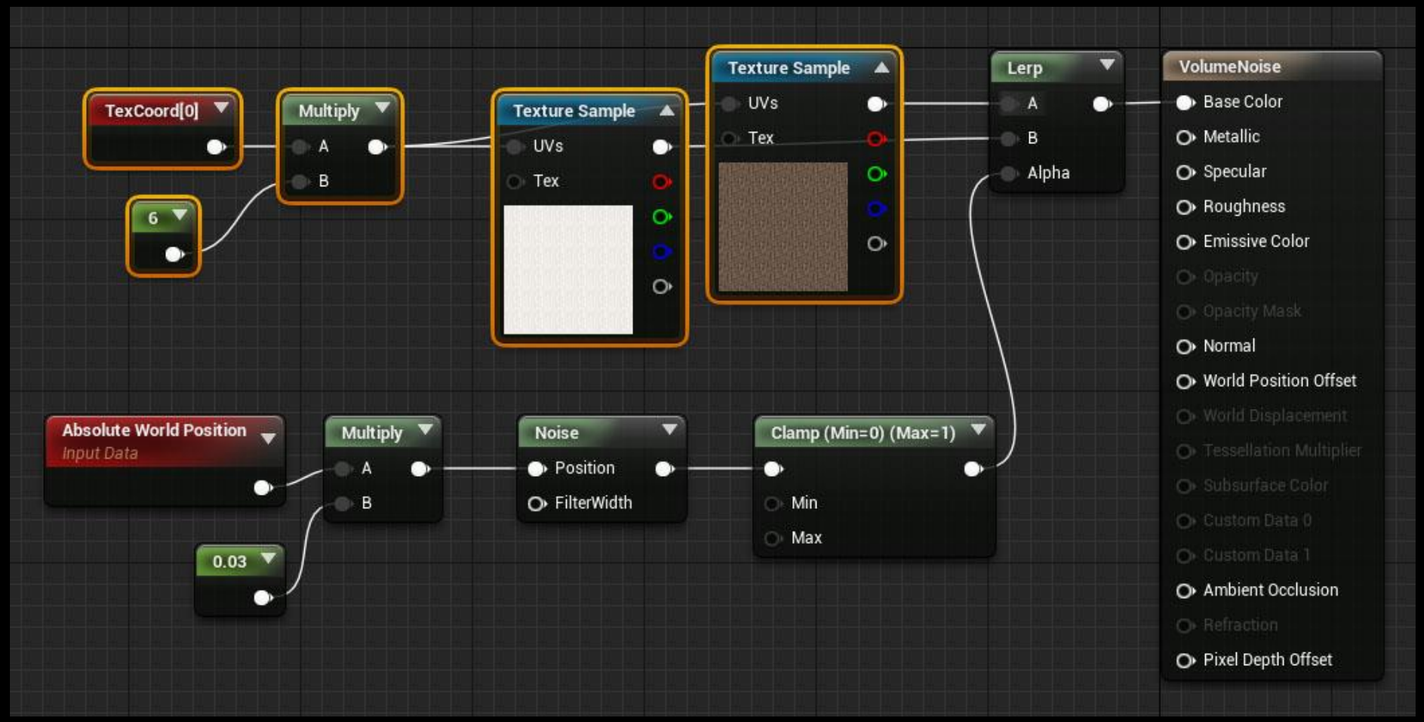
Noise Is Volumetric!



Use Noise As a Mask



Blend Between Textures

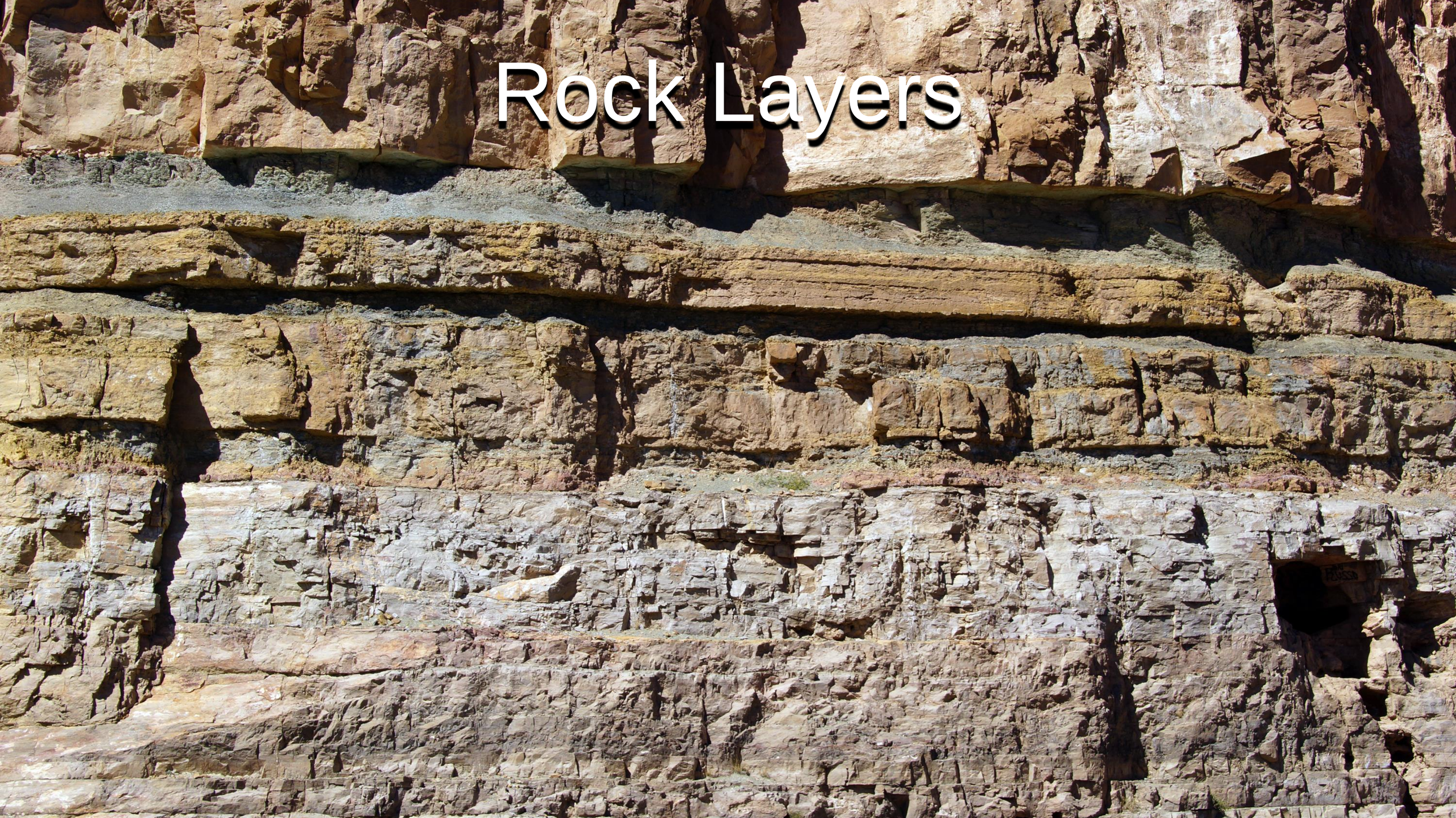


A Whole Forest of Unique Trees

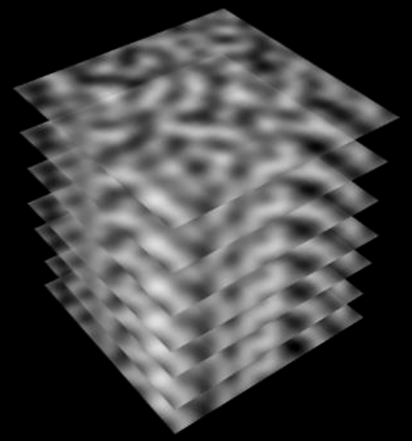
Noise Mask Code

```
float fastGradienClouds(float3 worldPos, int octaves, float lacunarity, float gain)
{
   float sum = 0, freq = 1.0, amp = 1.0;
   for(int i=0; i<octaves; i++)
   {
      sum += (tex3D(perlinNoiseVolumeTexture, position*freq).x * 2 - 1) * amp;
      freq *= lacunarity;
      amp *= gain;
   }
   return sum;
}
```

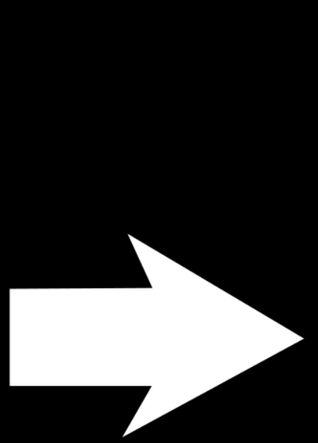
Rock Layers



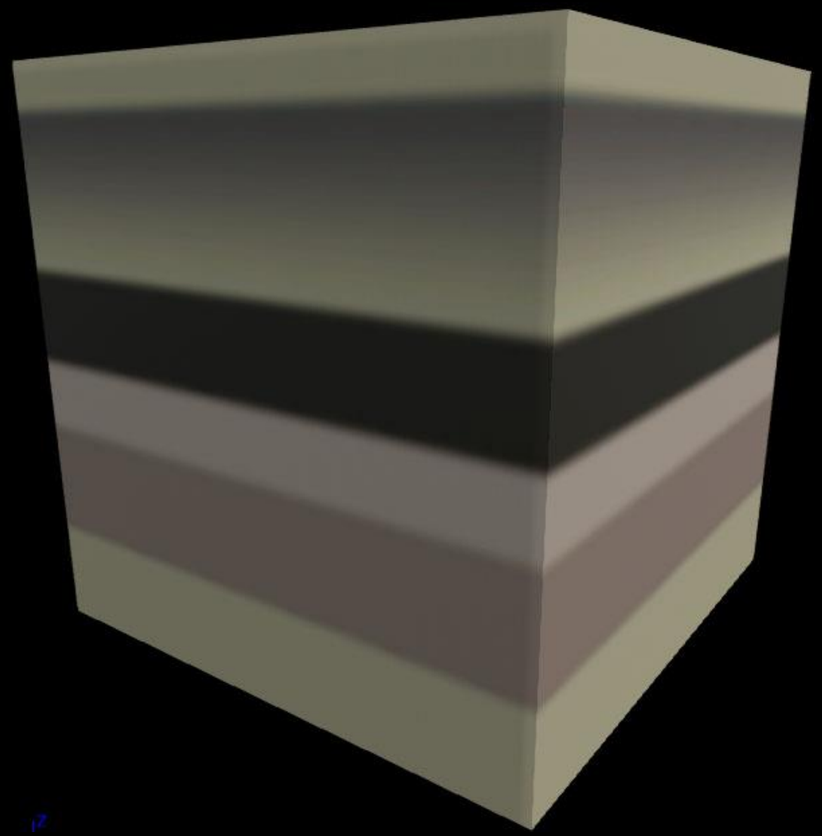
Gradient With Noise



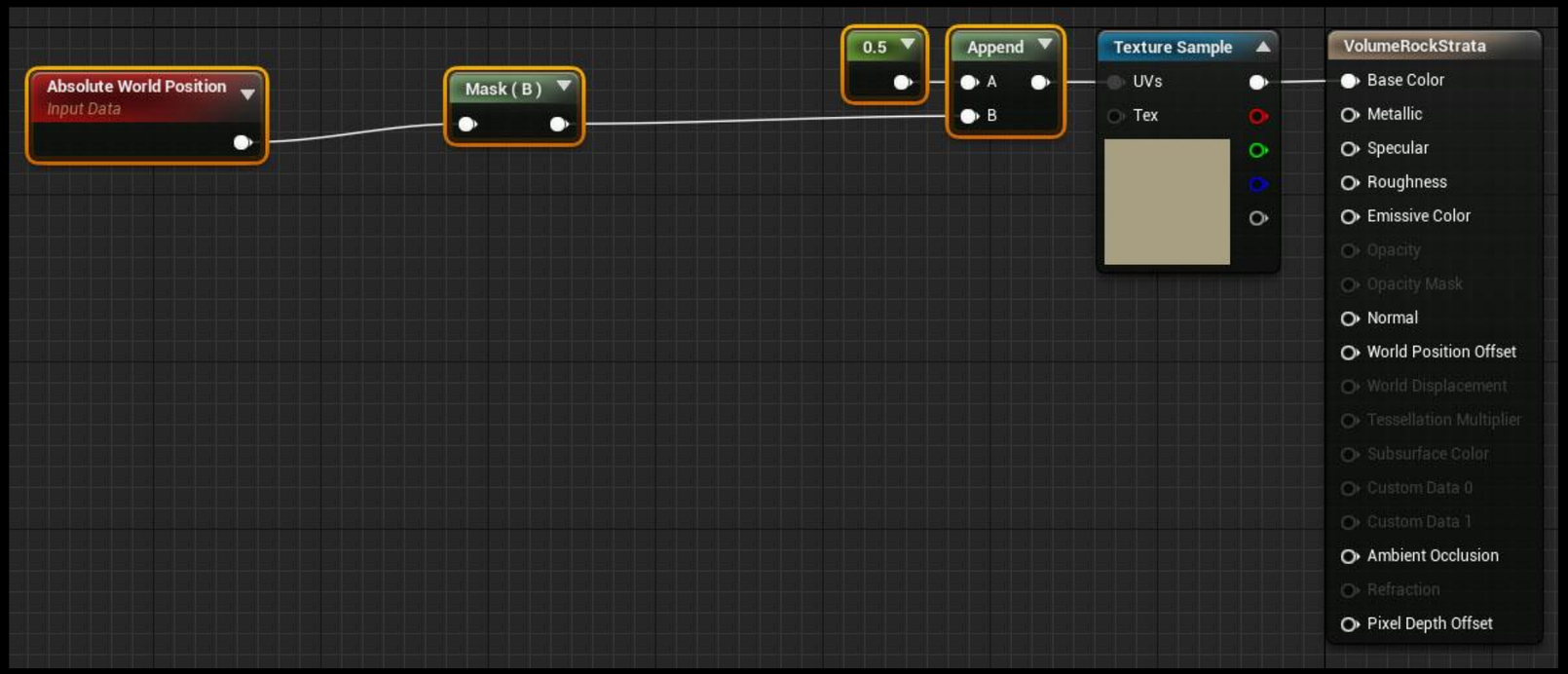
+ world Y position

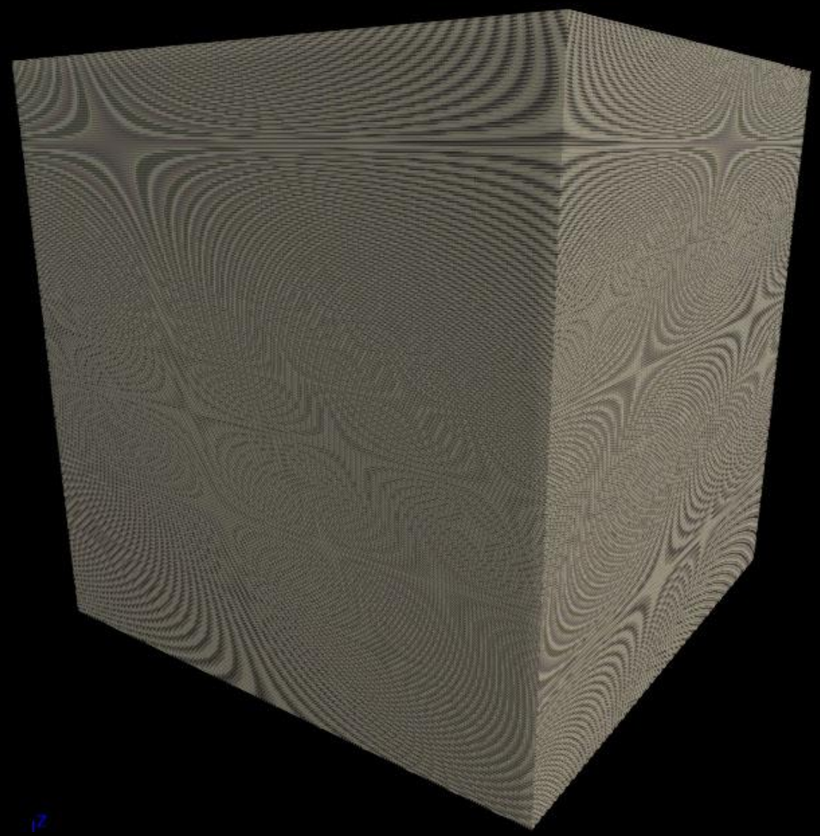


Start with Gradient

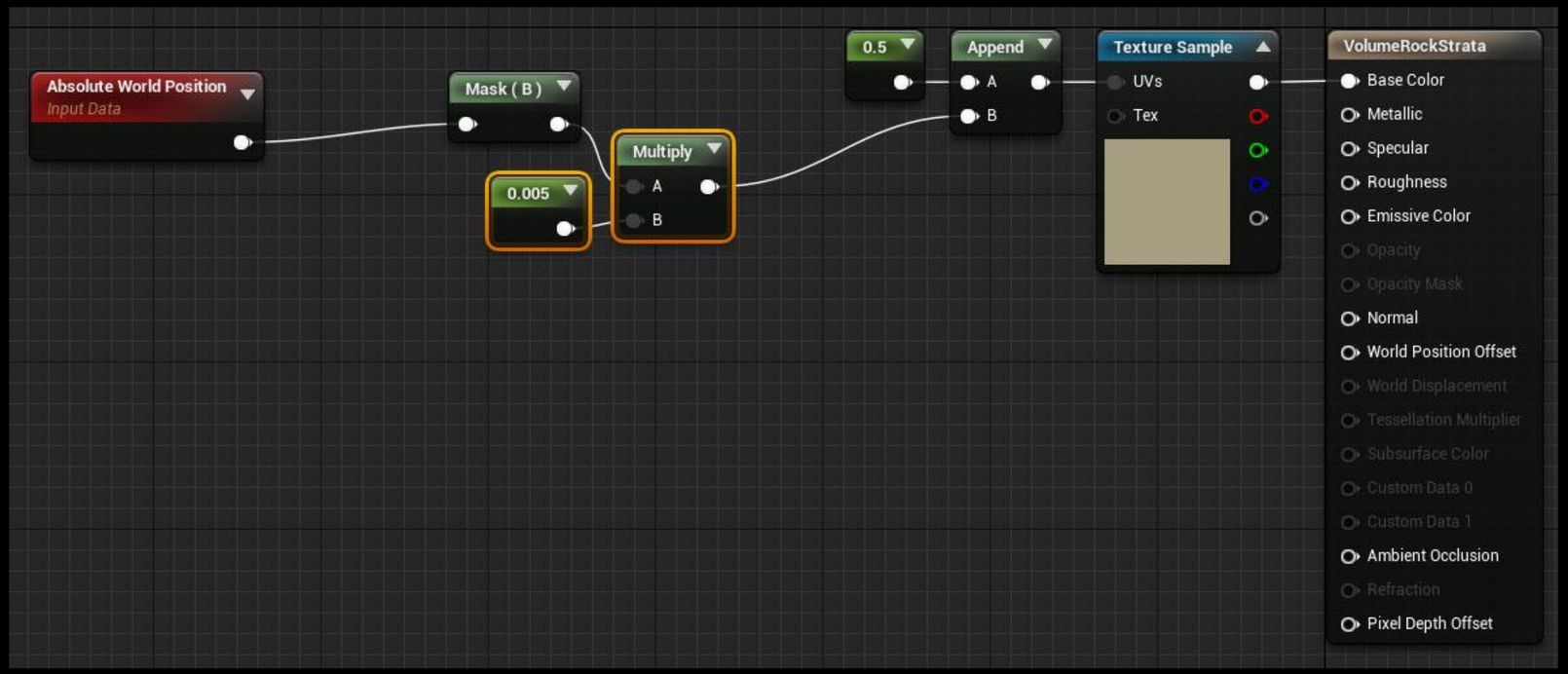


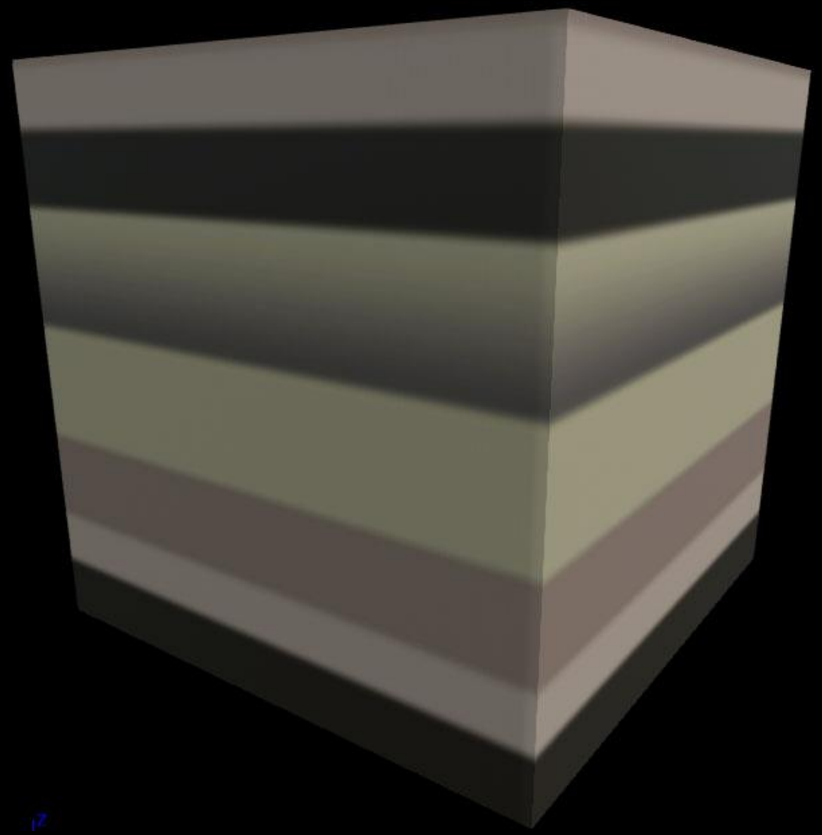
Sample with World Up



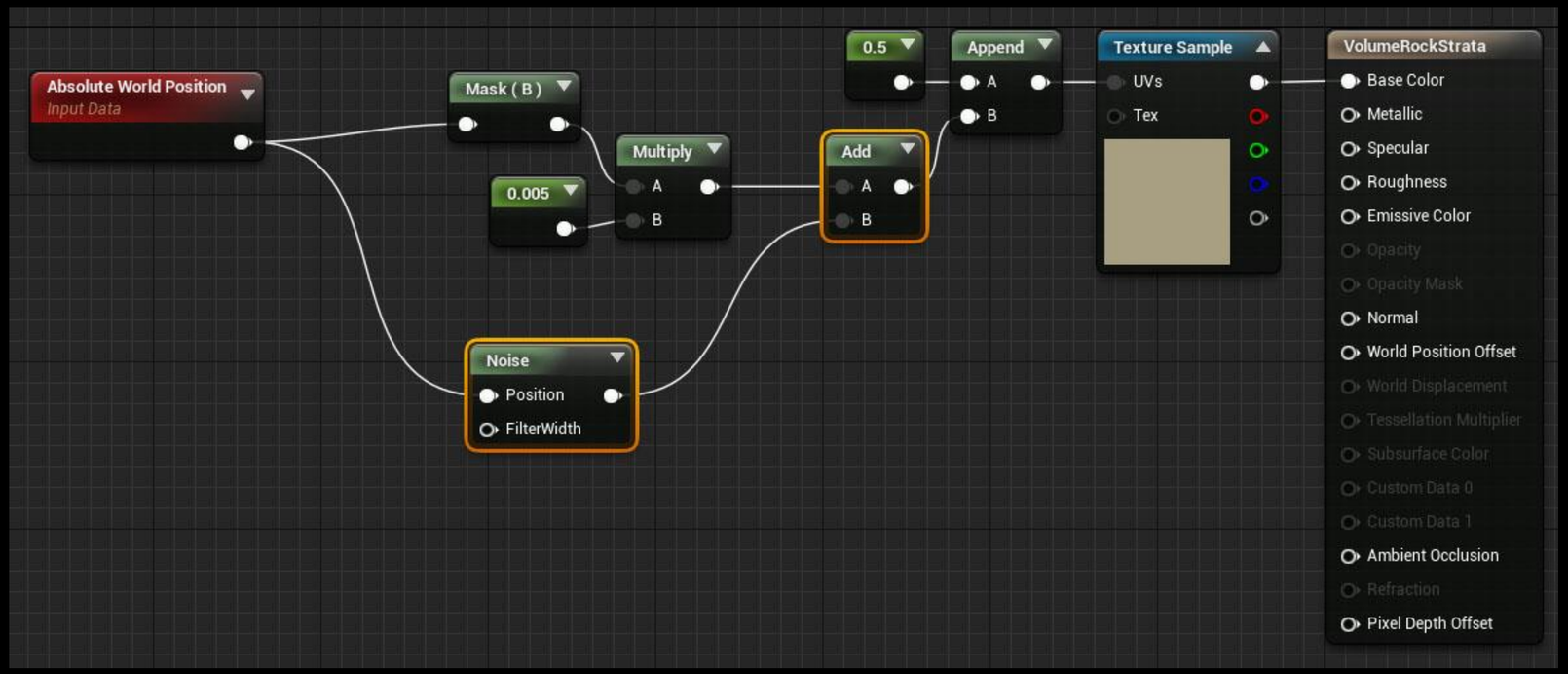


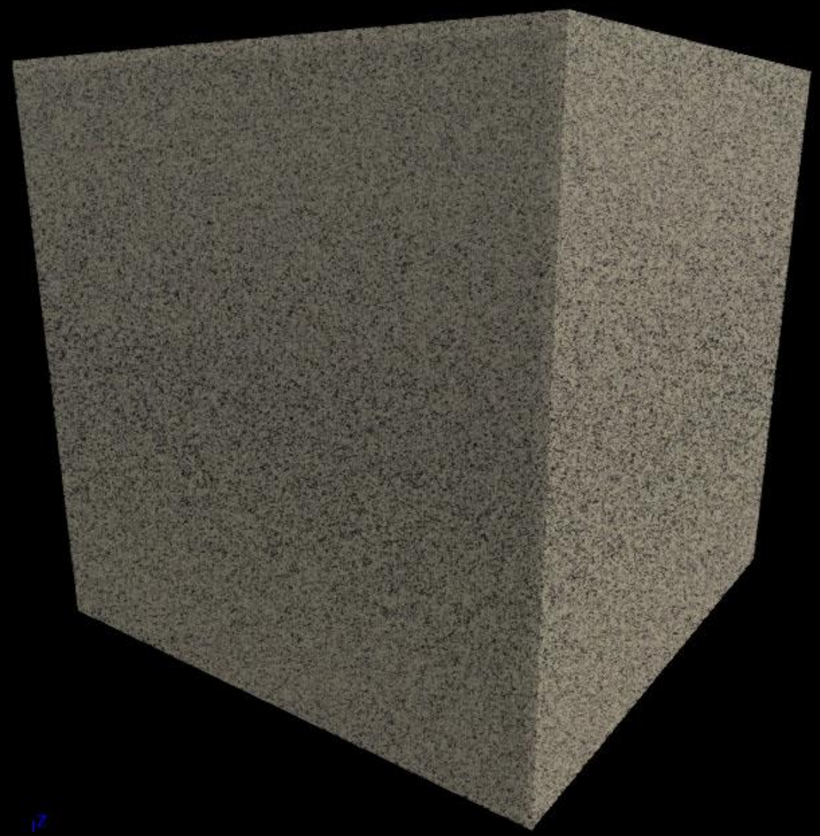
Scale World Space



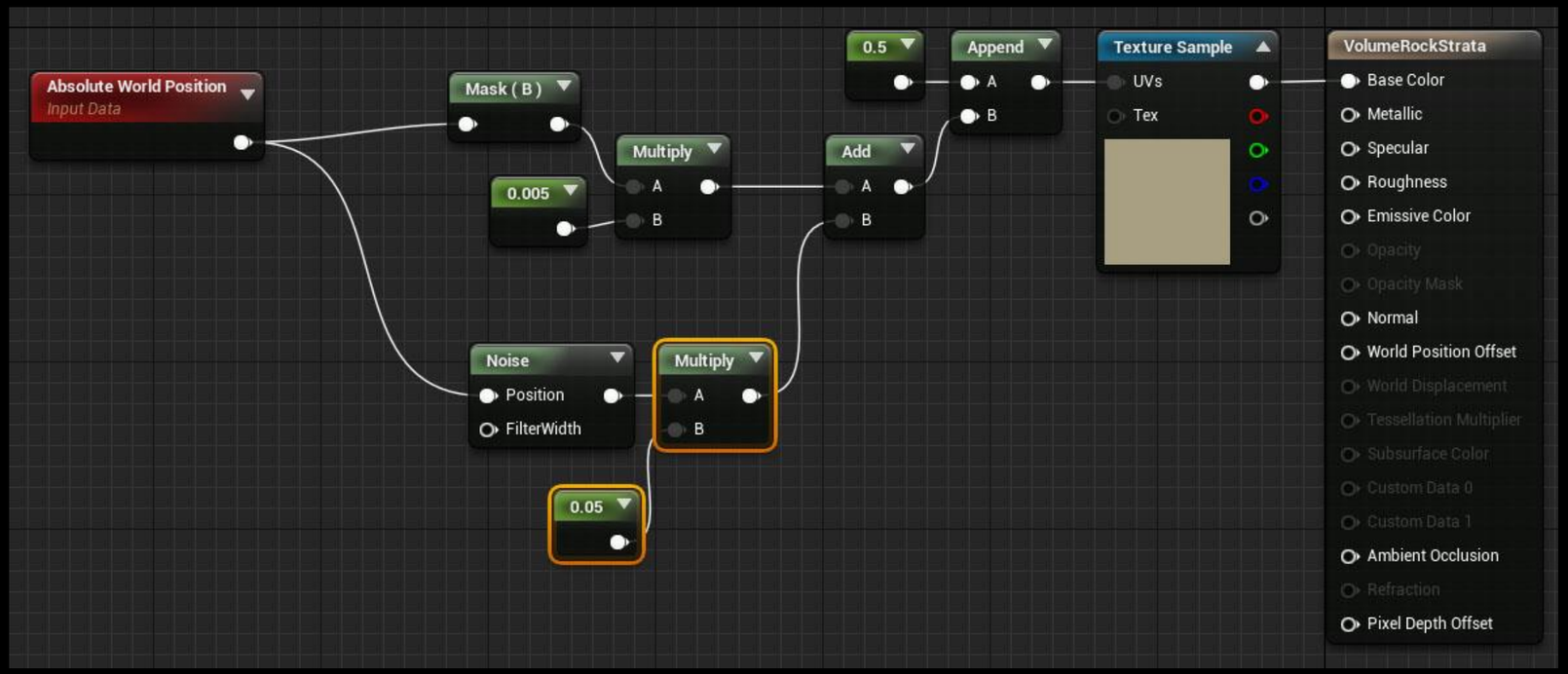


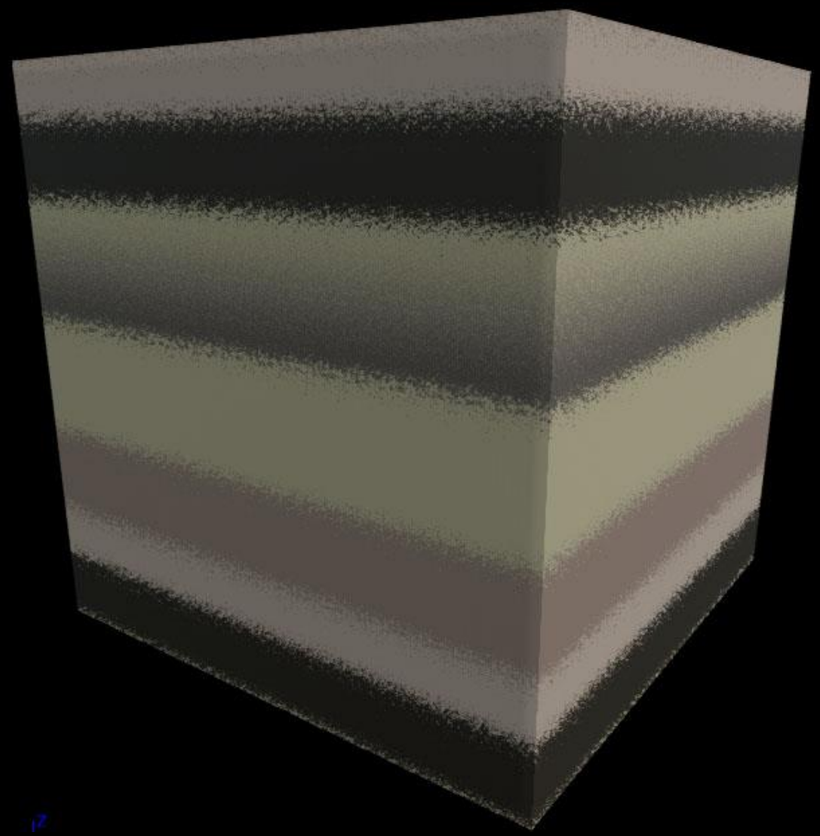
Scale World Space



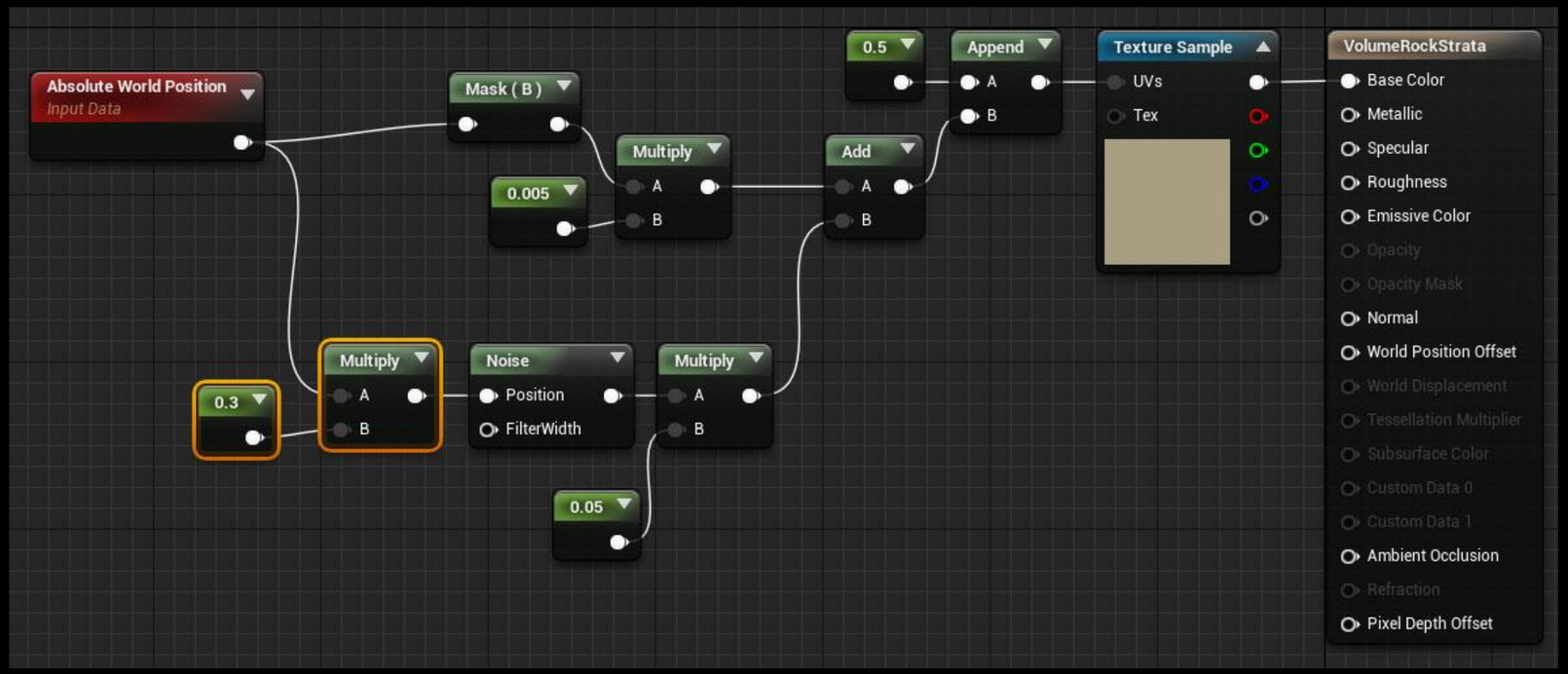


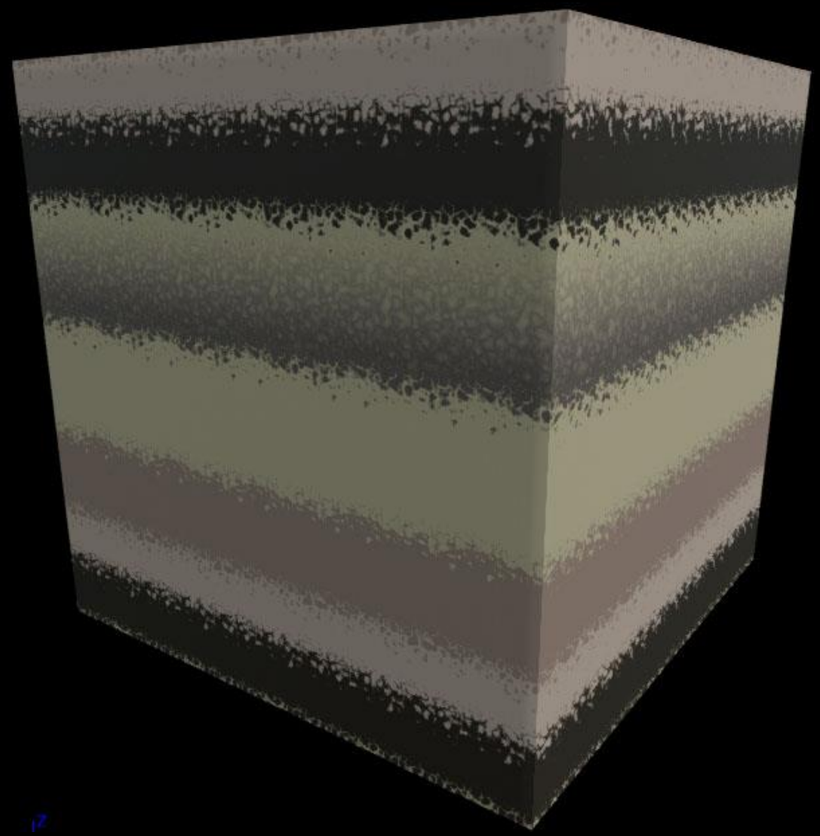
Tone Down the Noise

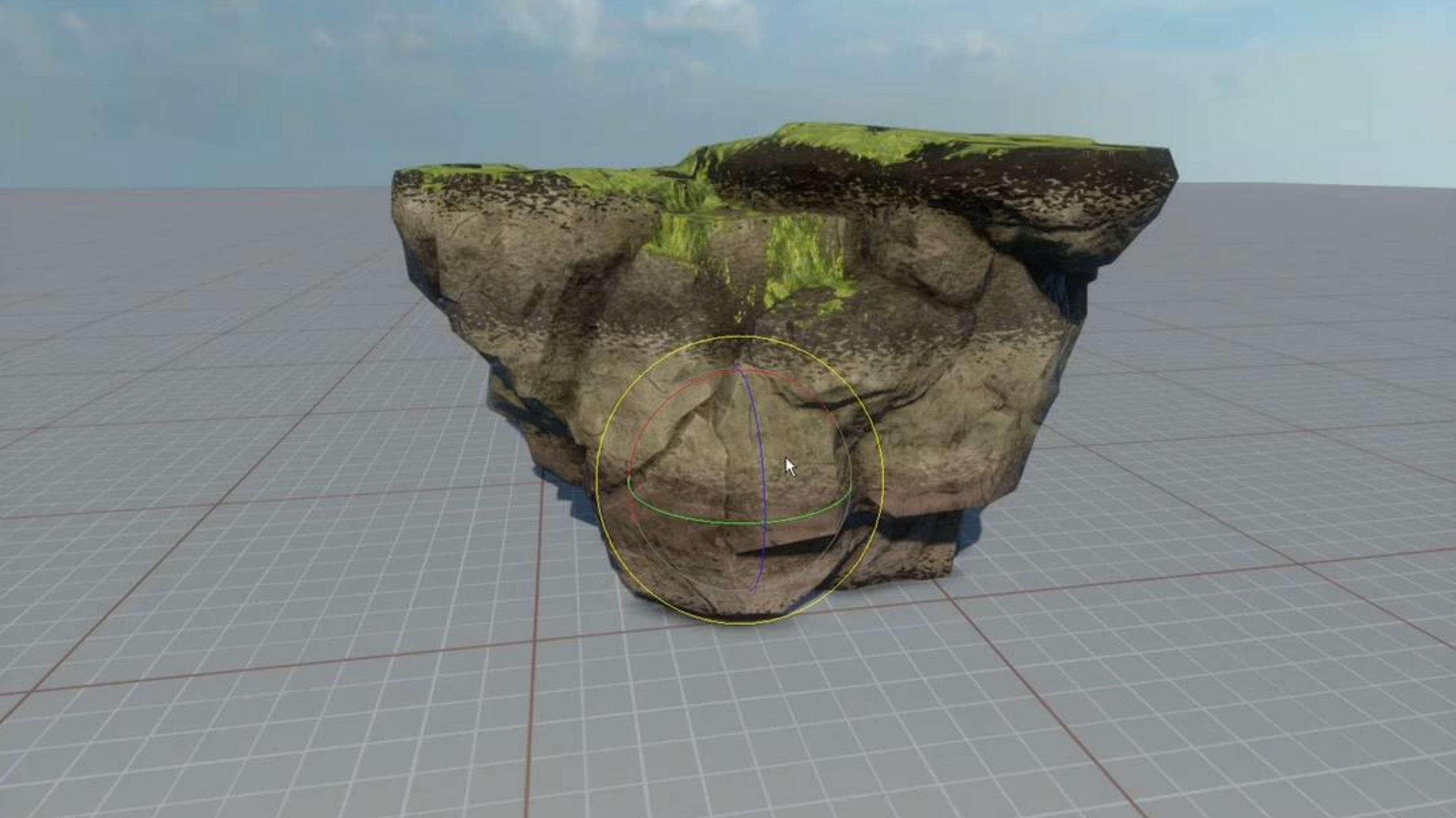




Scale the Noise







Rock Layers Code

```
float3 rockLayerColor(float3 worldPos, float scale)
{
    float scaledWorldY = worldPos.y * scale;
    float noise = fastGradienClouds(worldPos * 0.3, 4, 2, 0.5);
    float gradientYcoord = (noise * 0.05) + scaledWorldY;
    return tex2D(rockStrataGradient, float2(0.5, gradientYcoord)).rgb;
}
```

Next Steps

- Study what other games are doing
- Grab a book
- Google and YouTube Search
- Download Unreal or use ShaderFX in Max or Maya
- Write shaders!

Books!

- The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics - Randima Fernando
- Shaders for Game Programmers and Artists - Sebastien St-Laurent
- The COMPLETE Effect and HLSL Guide - Sebastien St-Laurent
- GPU Gems Series edited by Matt Pharr and Randima Fernando
- Shader X Series by Wolfgang Engel
- Advanced Lighting and Materials with Shaders - Kelly Dempski and Emmanuel Viale

Thanks!

ben@bencloward.com