Creative and Experimental VFX in The Last of Us Part II

Wataru Ikeda
VFX Artist - Naughty Dog

INTRODUCTION

- Wataru Ikeda (**Twitter: @WataruVFX**)
- About 17 years experience in Real-time VFX

  + 4 years as an Environment Artist

NAUGHTY DOG

ART Style: Photorealism

- Not open world
- Large VFX team
- Practical effects + techy effects

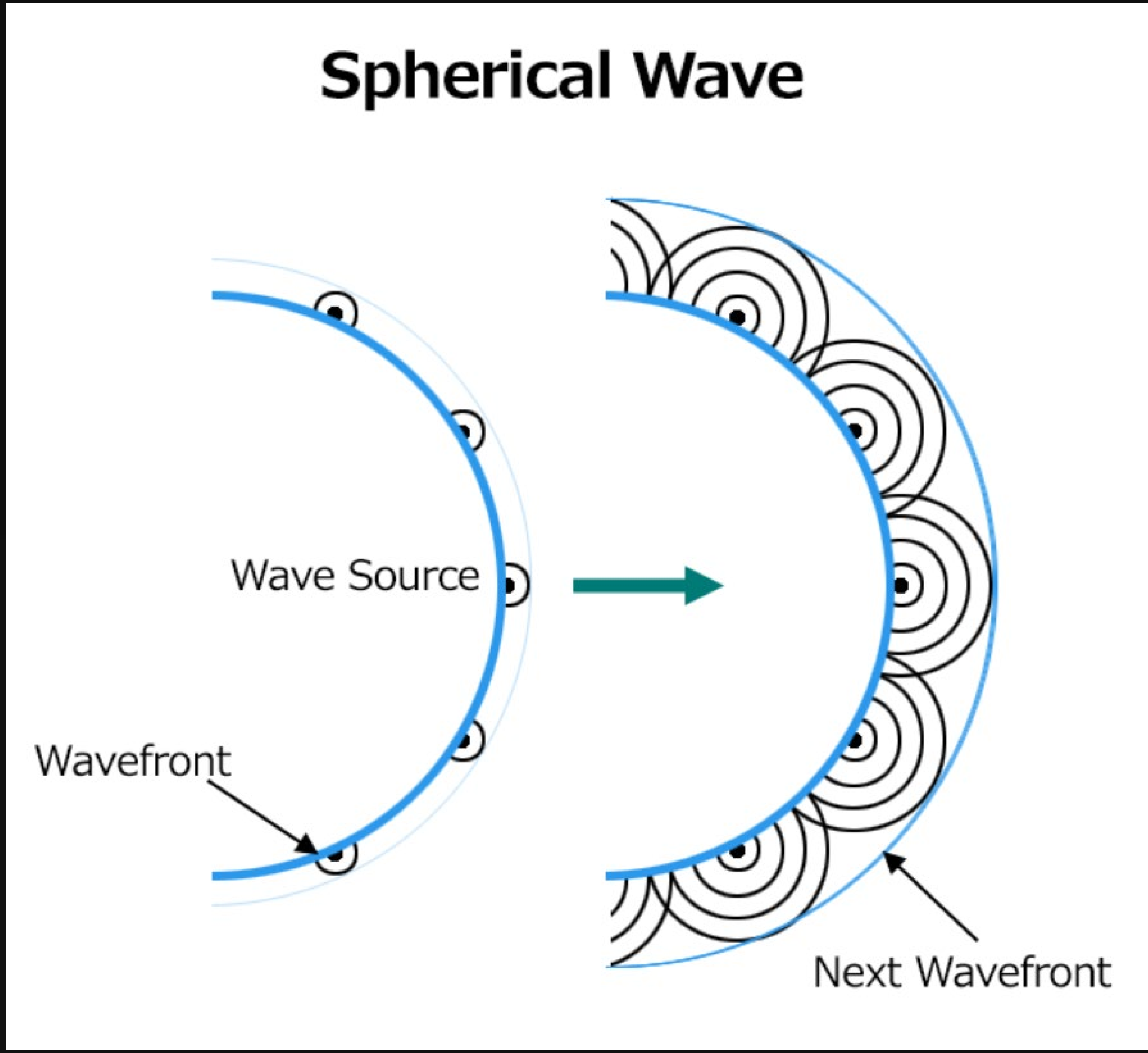
Contents

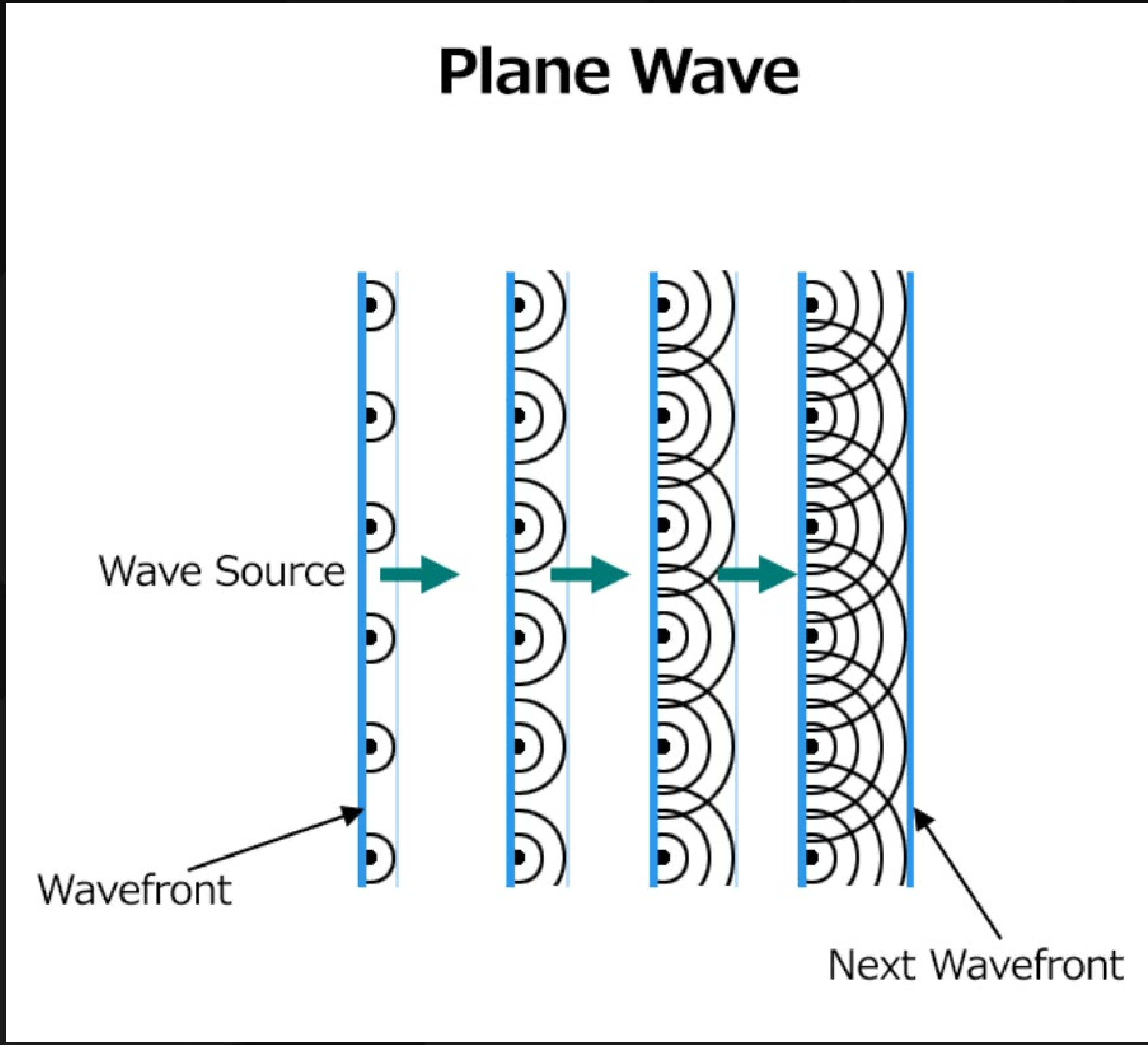
- Edge Ripples
- Curl Wave at Beach
- Updated DoF for Small Particles


- EDGE RIPPLES -

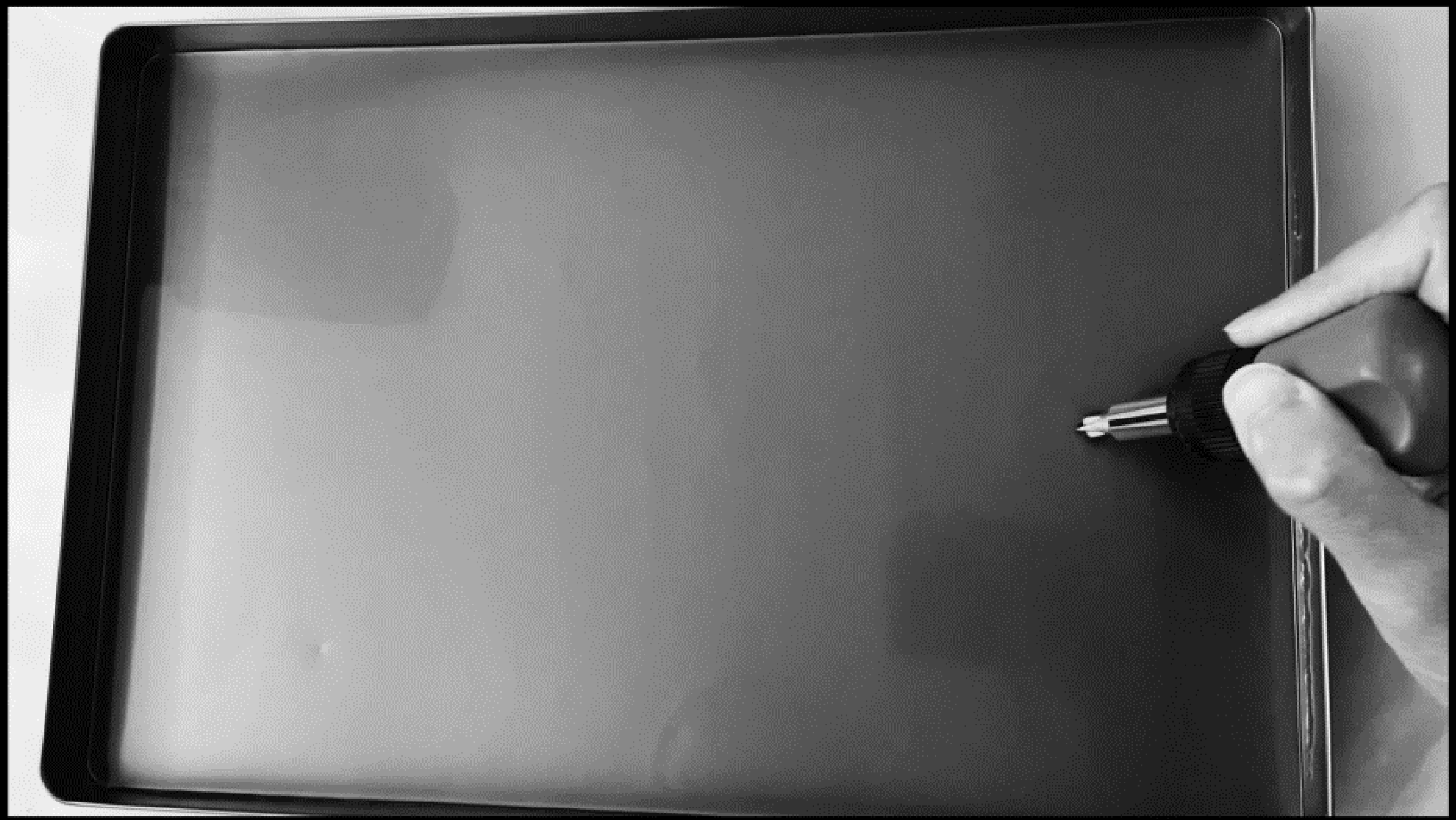
Reference: Mixkit

How and when ripples are produced?





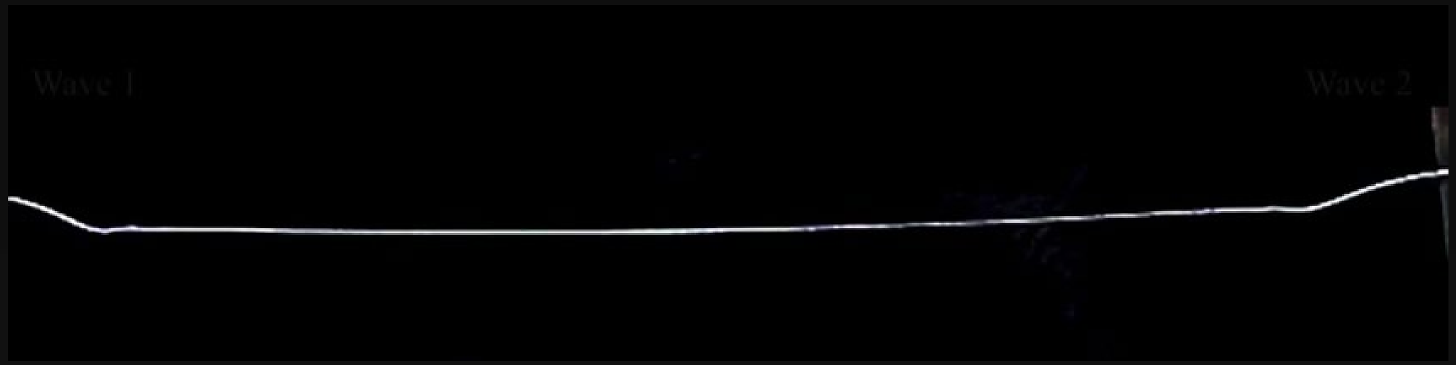
Observe not only the laws of physics, but also the actual video.



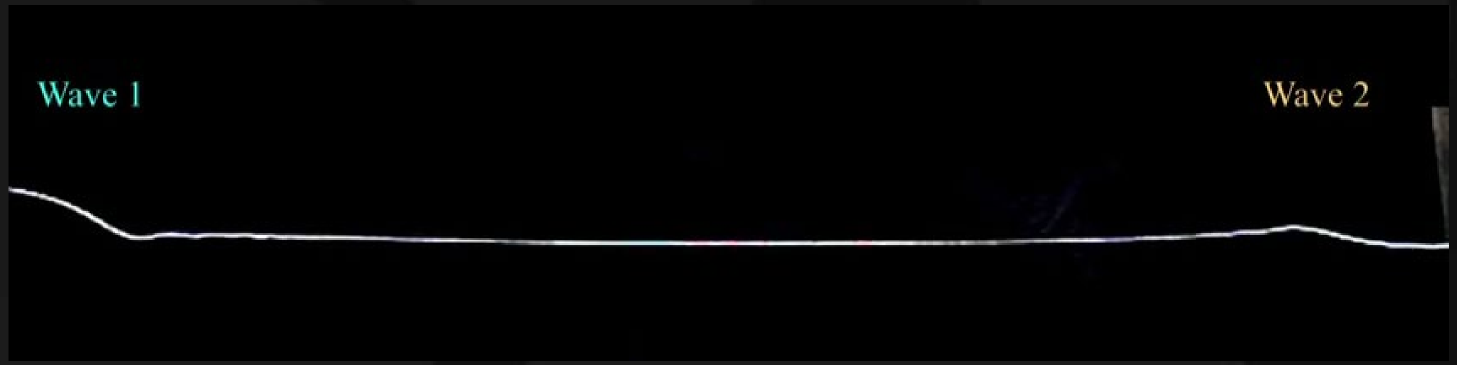
Spherical (Ripples) Waves

Plane Waves

Observe the Wave Interference



Constructive Wave; Wave1 + Wave2 = Double



Destructive Wave; Wave1 + Wave2 = Zero

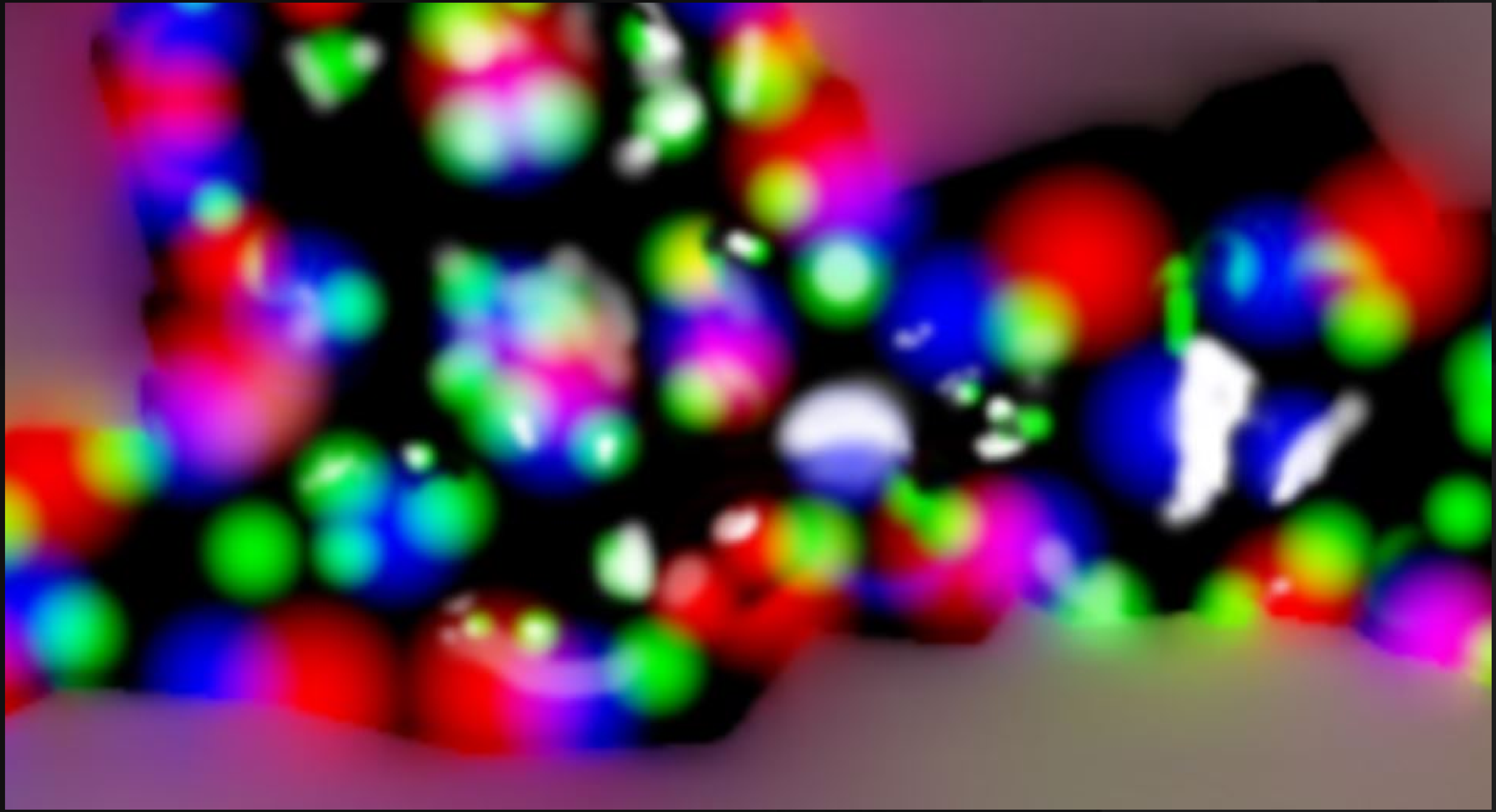
Reference:
“Show Me Some Science! Constructive and Destructive Interference”
CNS Little Shop of Physics (Colorado State University)

Challenge

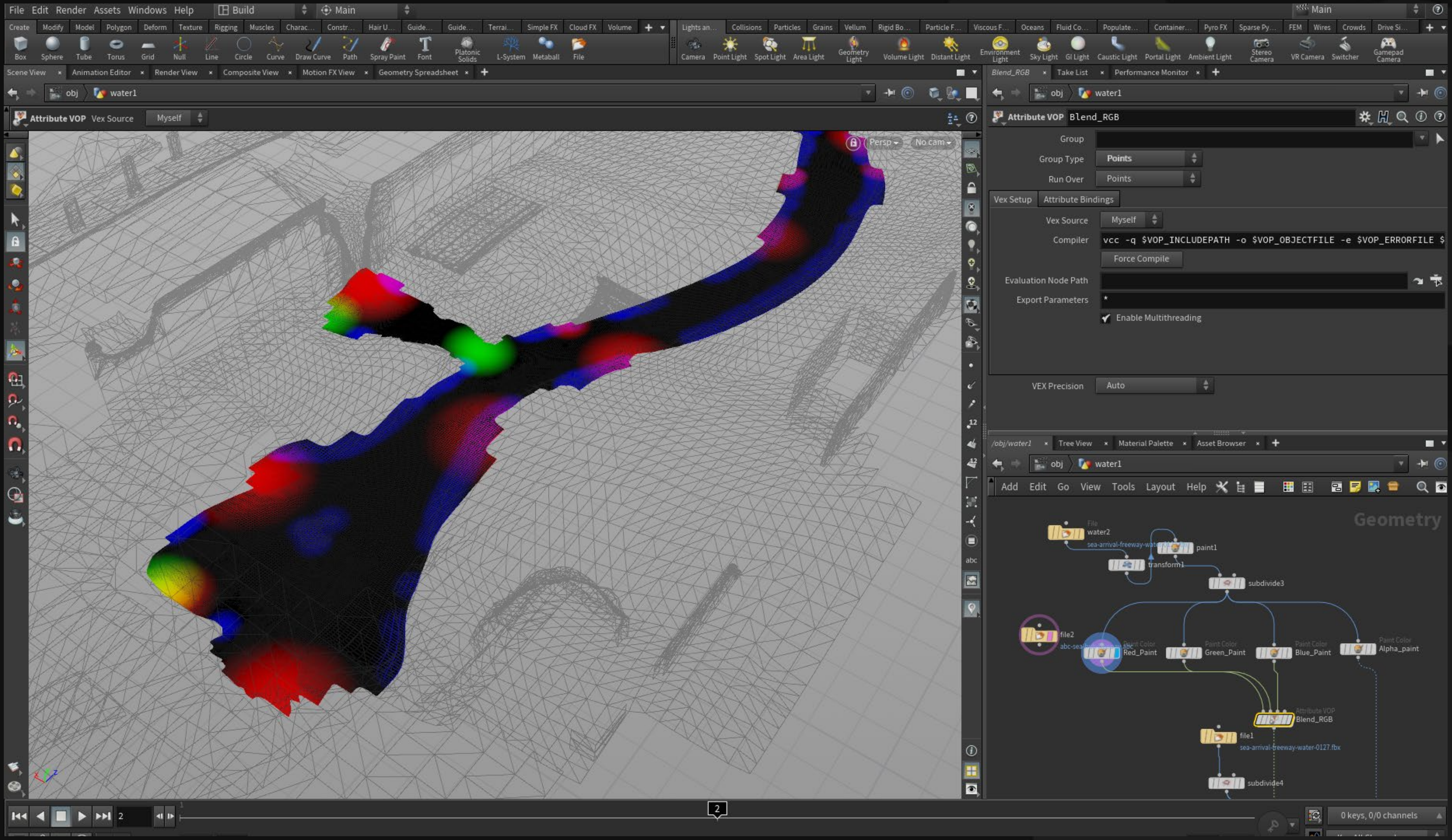
- Adding those complex ripples as much as we can!
- Do it cheap!

Generate Ripples using a painted texture

3 type of ripples(RGB) and mask(A= white color on this picture)



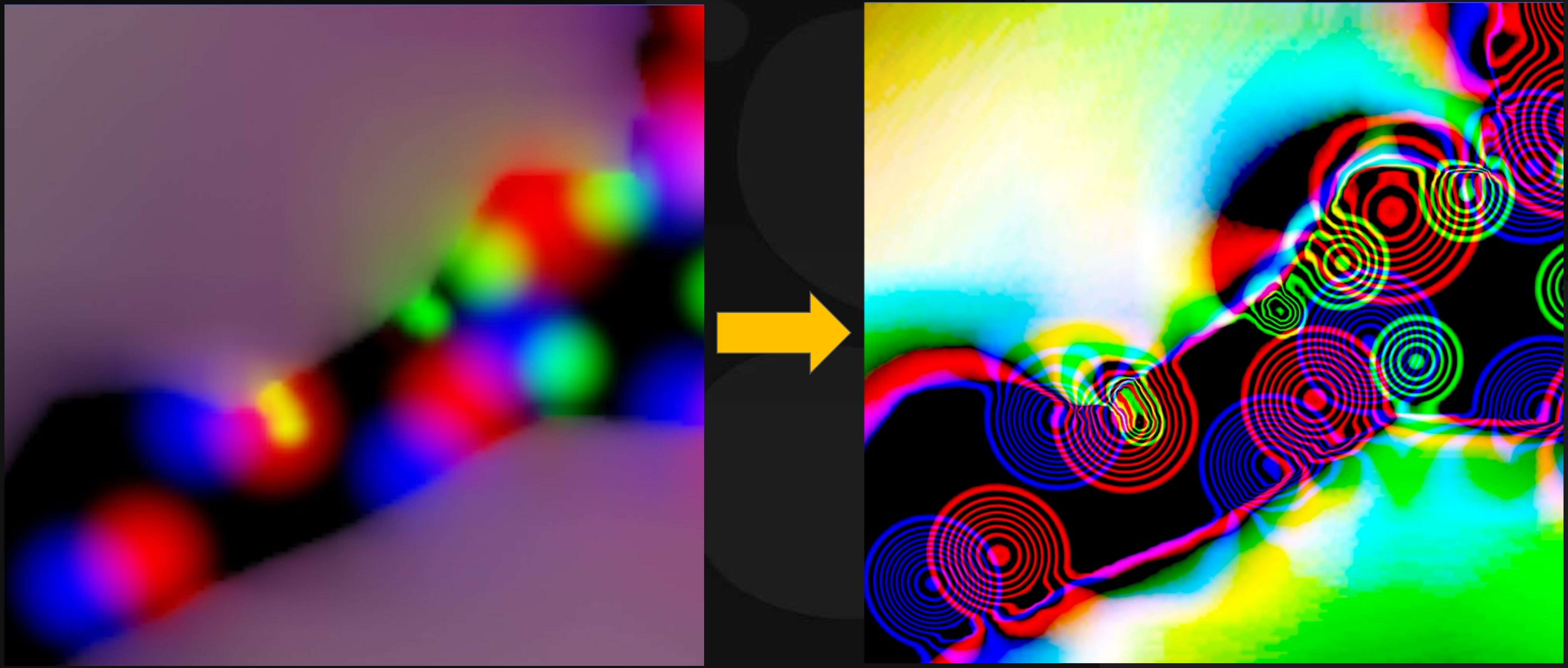
Generate Ripples using a painted texture



Convert from Texture's gradient to Sine Wave

Like the following shader code…

```
float3 ripplewaves=
 sin(SampledTexture.rgb * RIppleIteration + time * timespeed);
```



Convert from Texture's gradient to Sine Wave

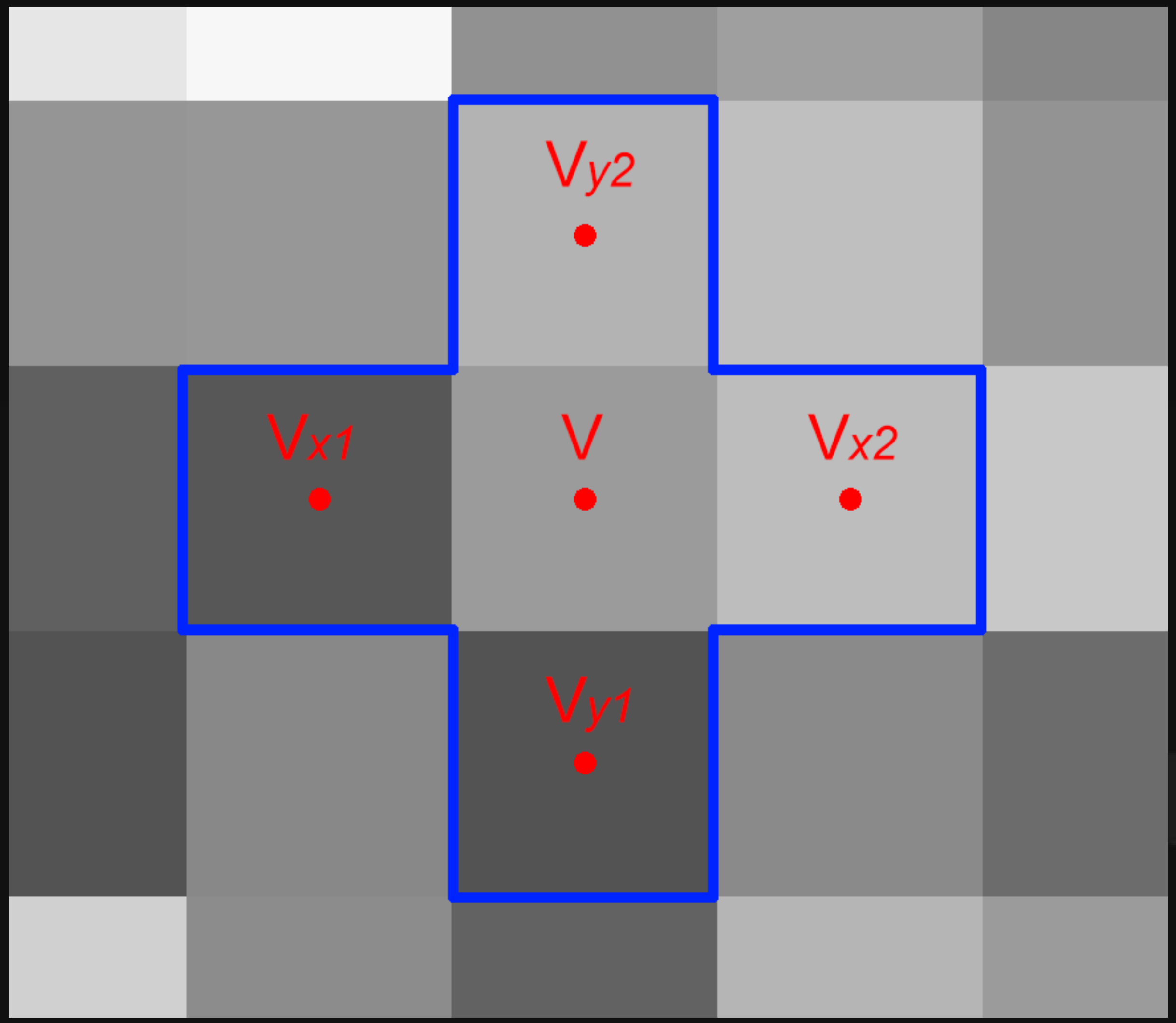
```
float3 ripplewaves =
sin(SampledTexture.rgb * RIppleIteration + time * timespeed);
```

A trick of Generating Normal

Trying to get BETTER and FASTER…

Common Generating Normal Map from Height

4 Textures sampling & 2 cross products



du=V*x*2−V*x*1, *dv*=V*y*2−*Vy*1

$$\overrightarrow{T_x} = (1,\ 0,\ du),\ \overrightarrow{T_y} = (0,\ 1,\ dv)$$

$$\vec{N} = \overrightarrow{T_x} \times \overrightarrow{T_y} = (-du,\ -dv,\ 1)$$

Common Generating Normal Map from Height

- It's a cheaper way than the previous slide. This allows to make a normal map.

```
float4 sampleda = tex2D(Sampler, UV);
float4 sampledb = tex2D(Sampler, UV + float2(g_UVOfs.x,0);
float4 sampledc = tex2D(Sampler, UV + float2(0, g_UVOfs.y);
float dota = dot(samplea, float3(1,0,0));
float dotb = dot(sampleb, float3(1,0,0));
float dotc = dot(samplec, float3(1,0,0));
float2 difa = float2((dotb – dota) * g_normalintensity, 0);
float2 difb = float2(0, (dotc – dota) * g_normalintensity);
float3 normal = cross(difa, difb);
```

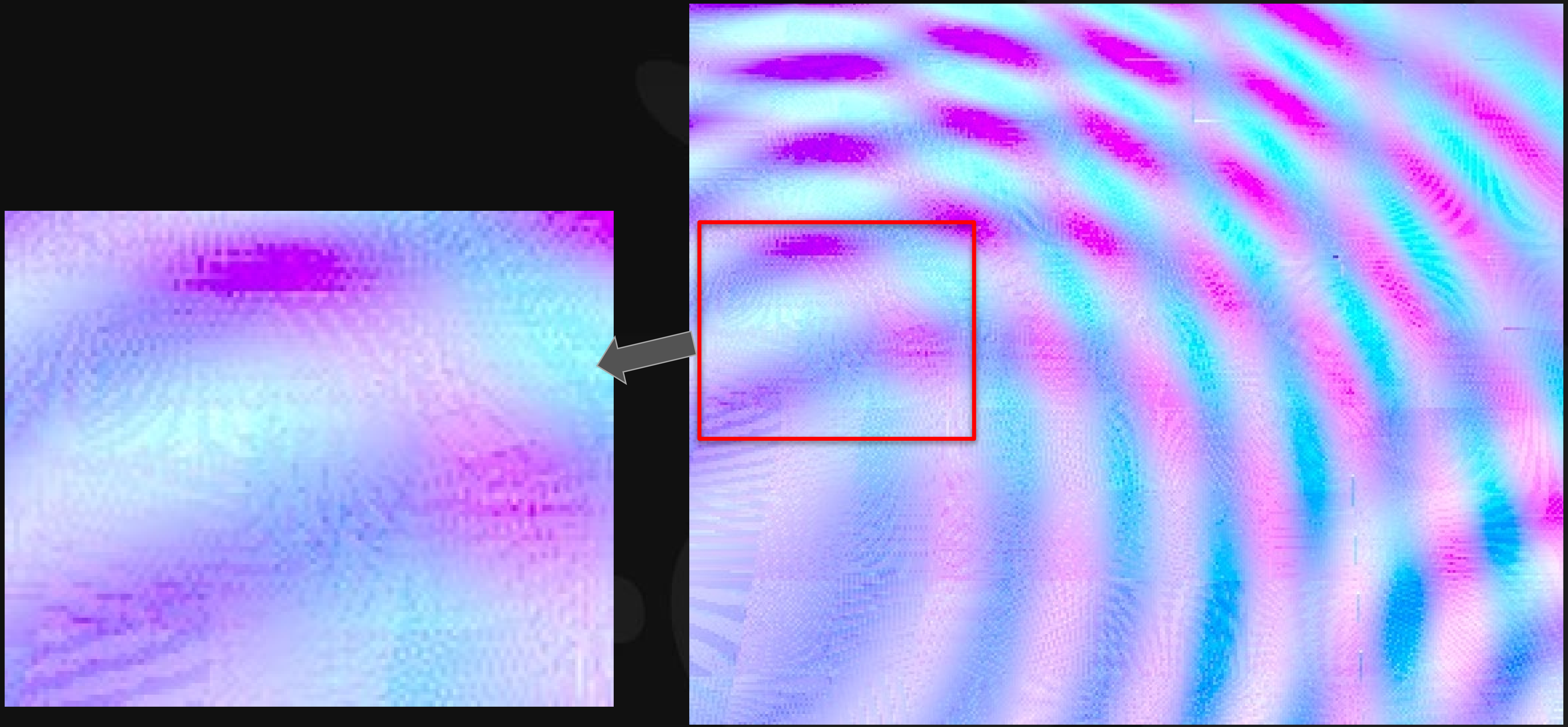
Common Generating Normal Map from Height

- using DDX & DDY

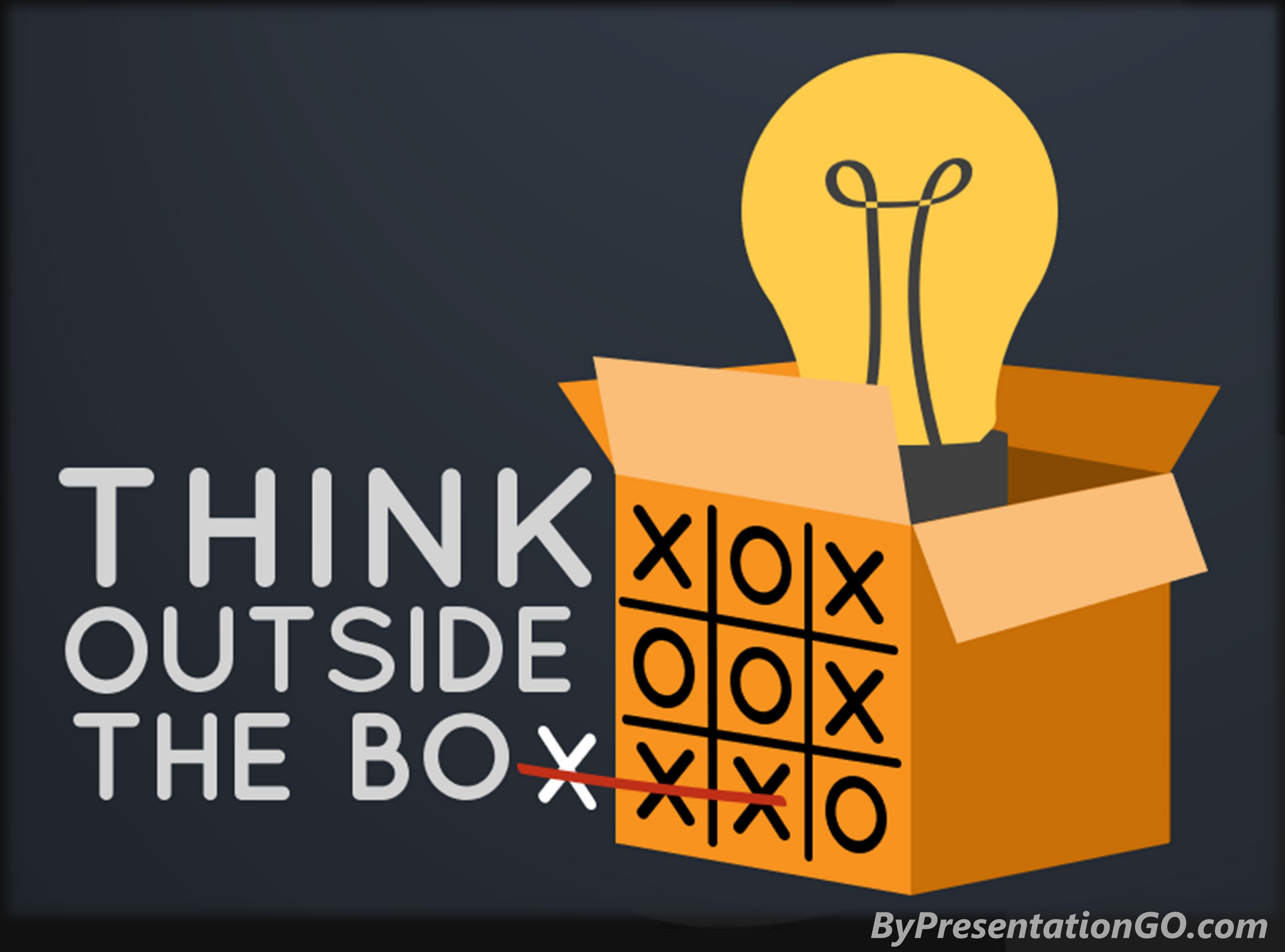
```
float3 ddxs = ddx(sampledTex);
float3 ddys = ddy(sampledTex);
float3 normals = float3((ddxs.x + ddxs.y + ddxs.z) / 3.0,
                        (ddys.x + ddys.y + ddys.z) / 3.0,
                        1.0) * 0.5 + 0.5;
```

Common Generating Normal Map from Height

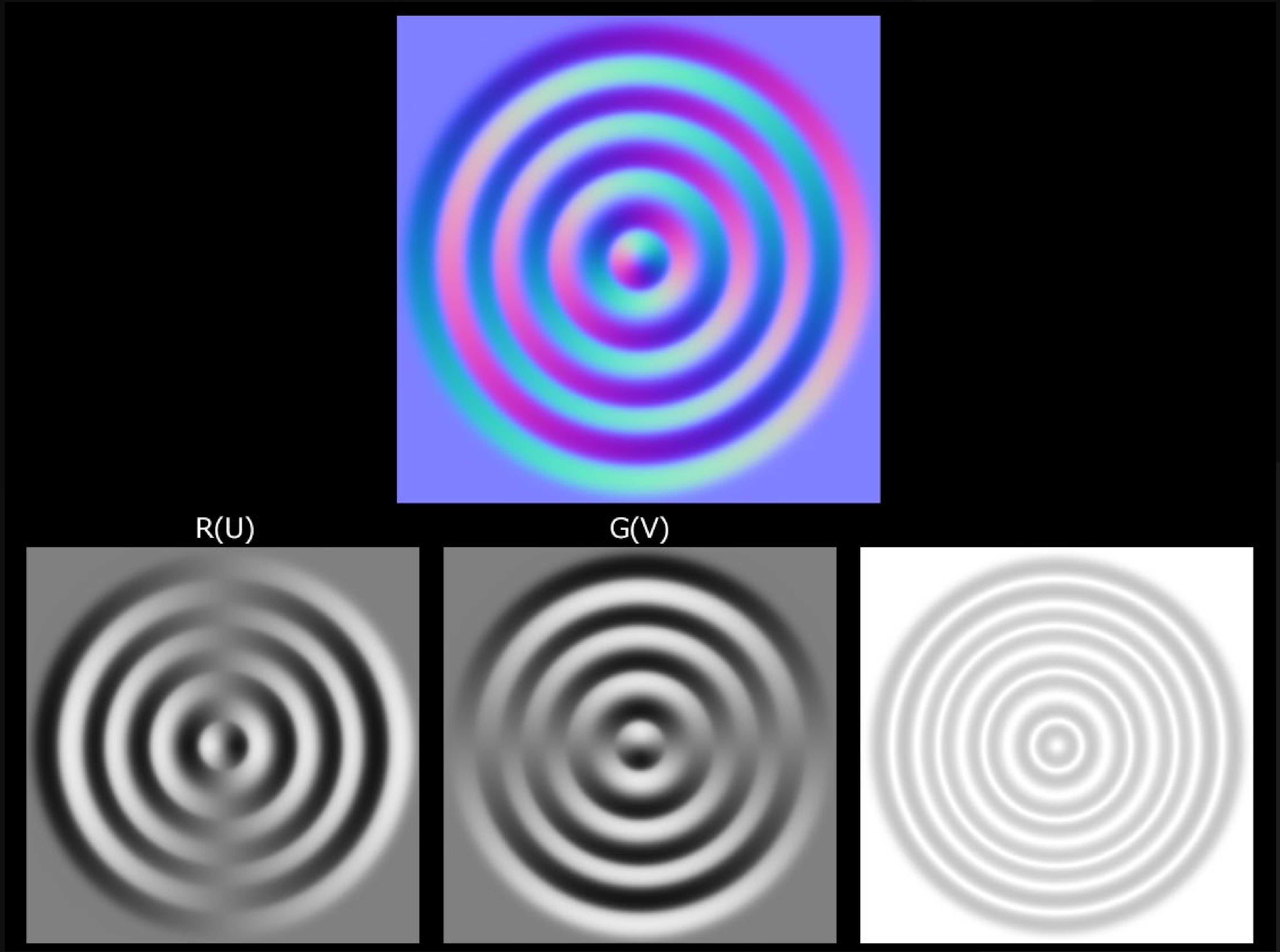
- using DDX & DDY often has noise.



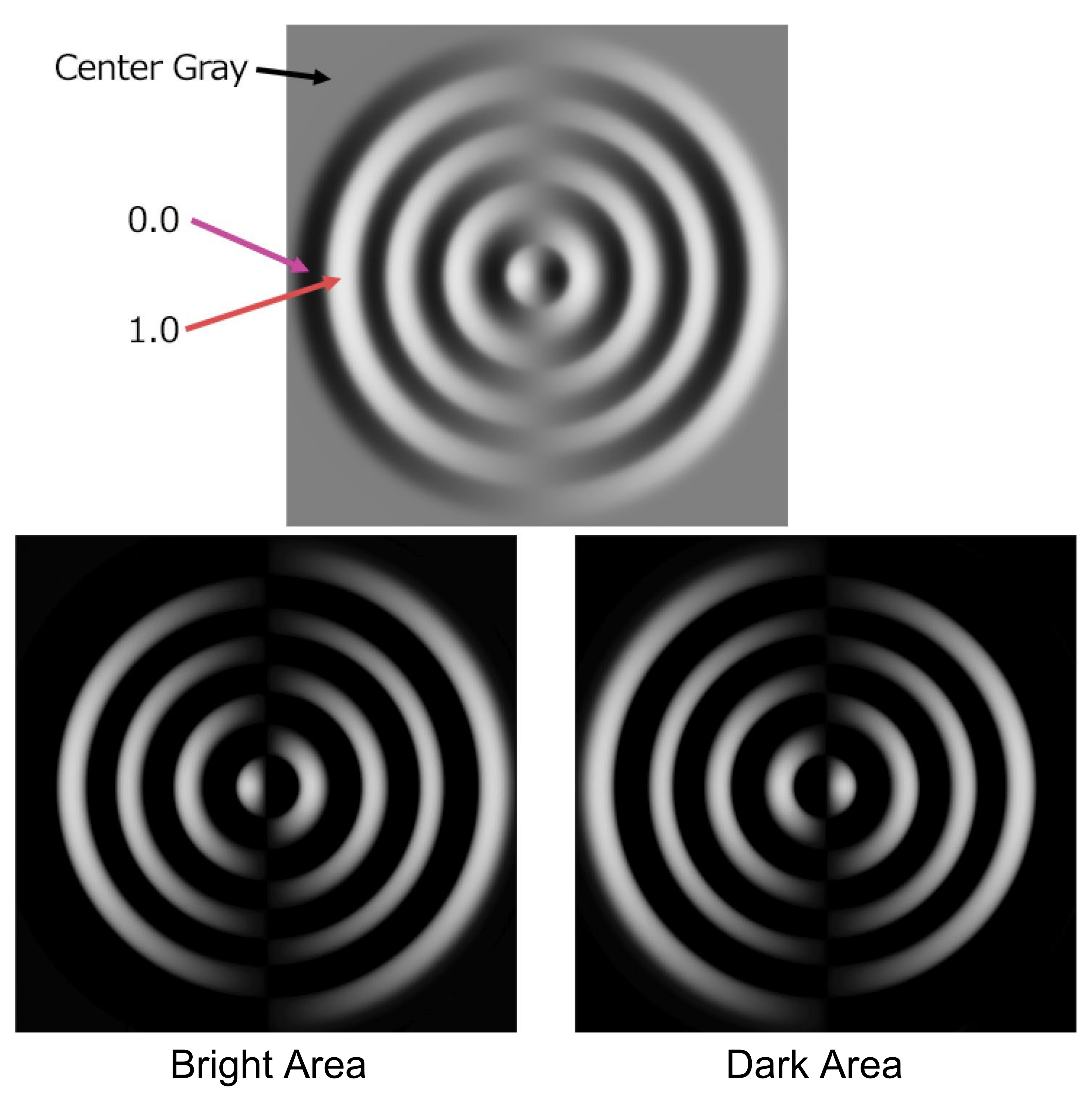
Changing The Perspective



Observation of Normal Map

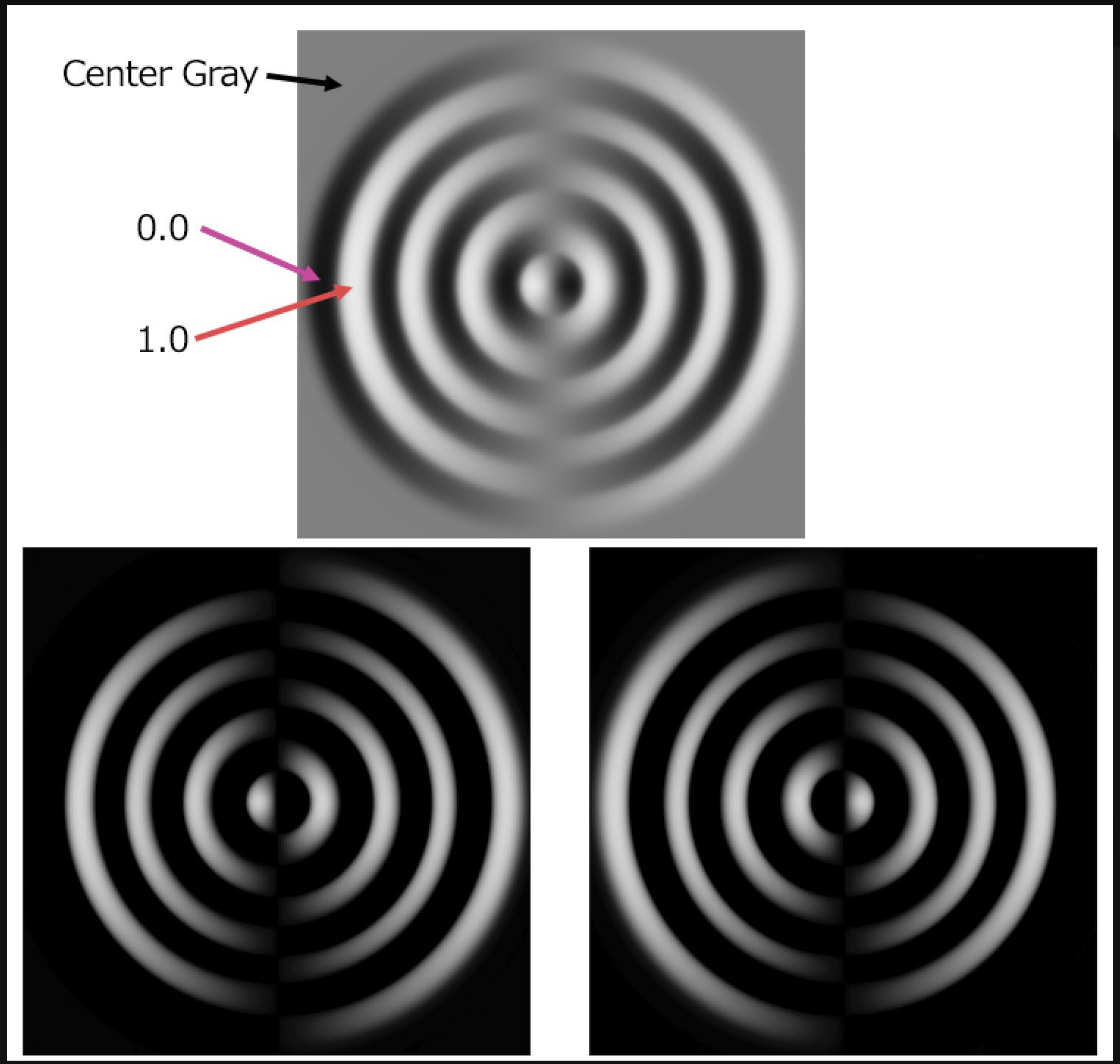


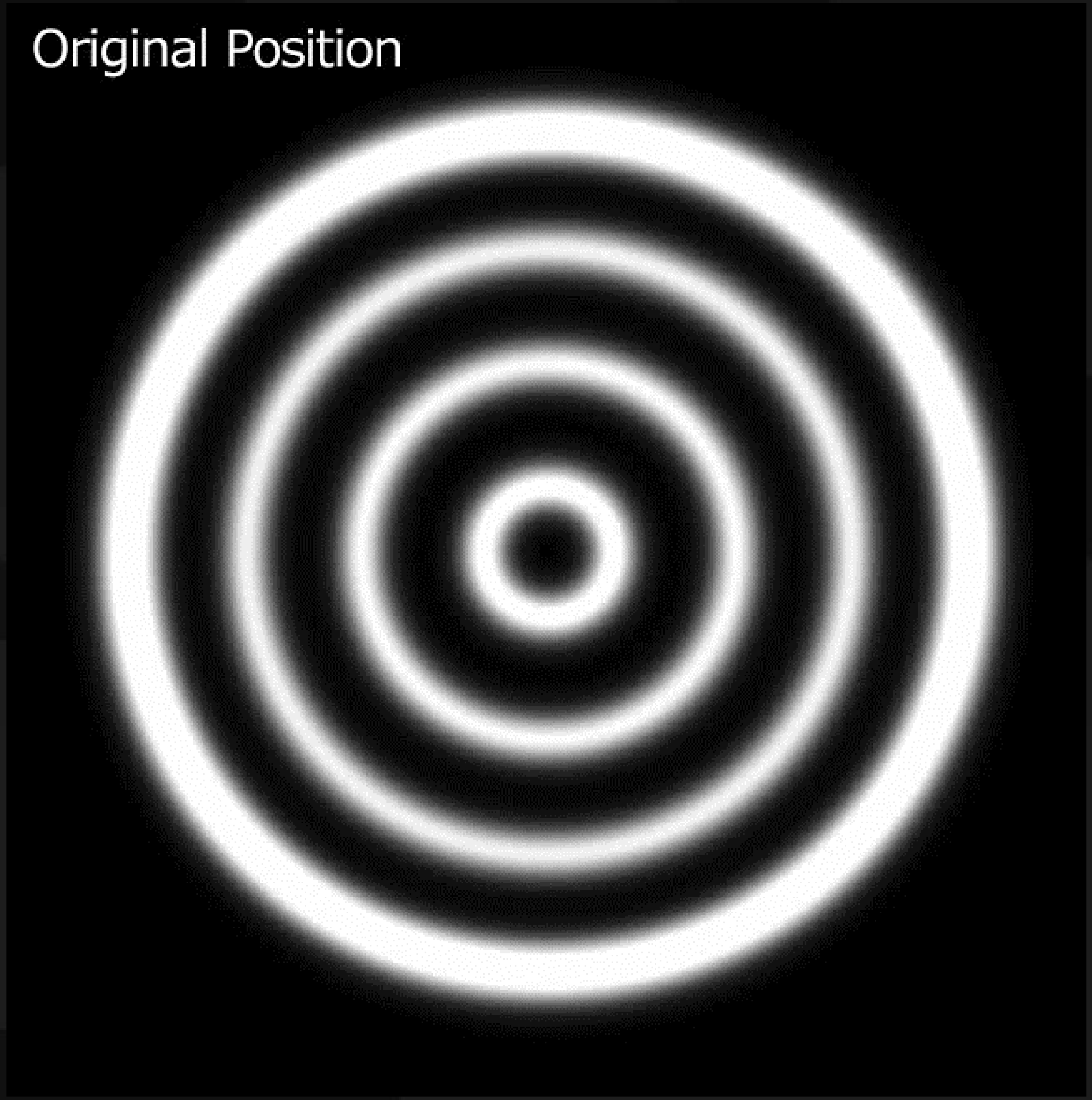
Observation of Normal Map



Observation of Normal Map

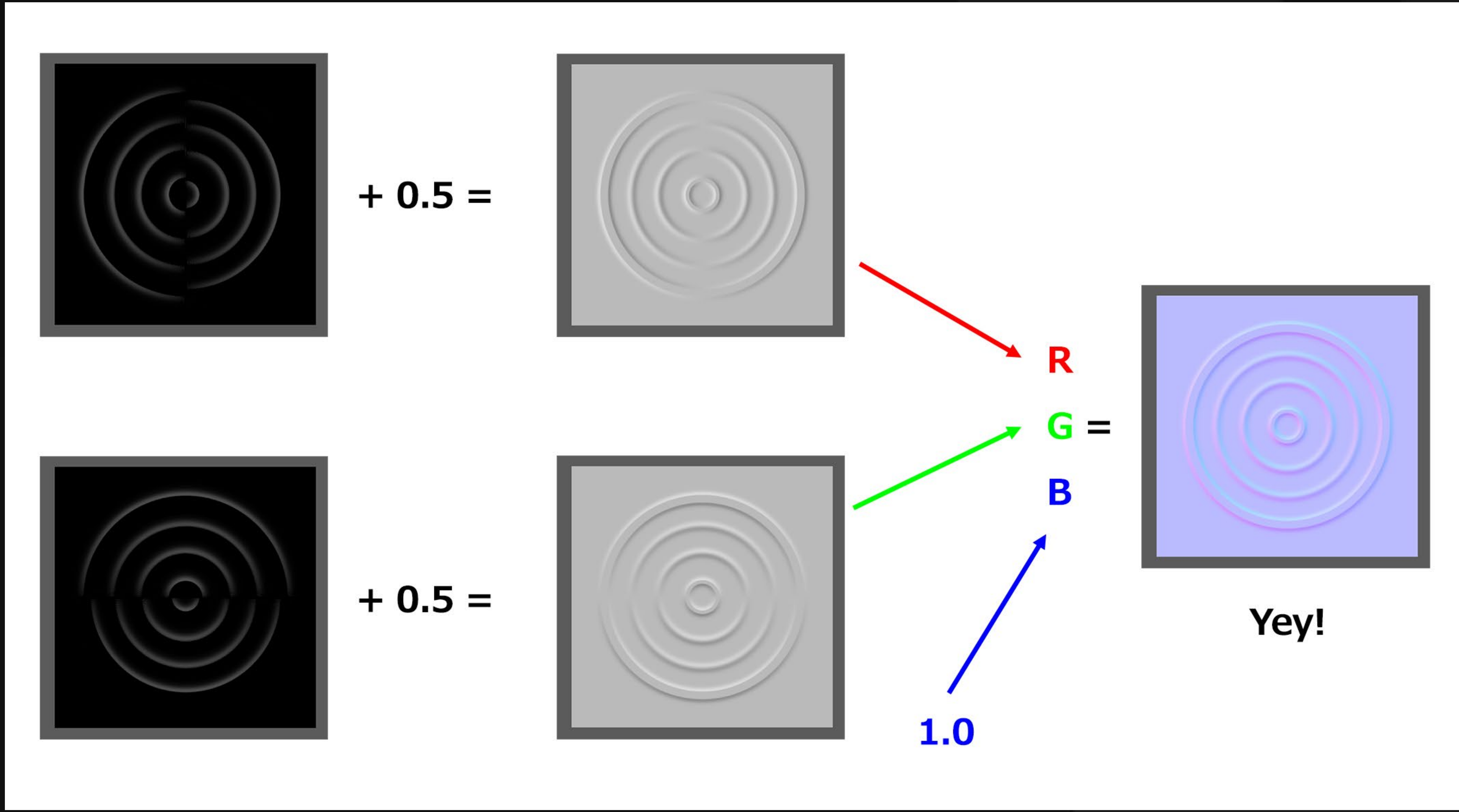
Making bright and dark areas ≒ Subtraction between Original and Offset Position.





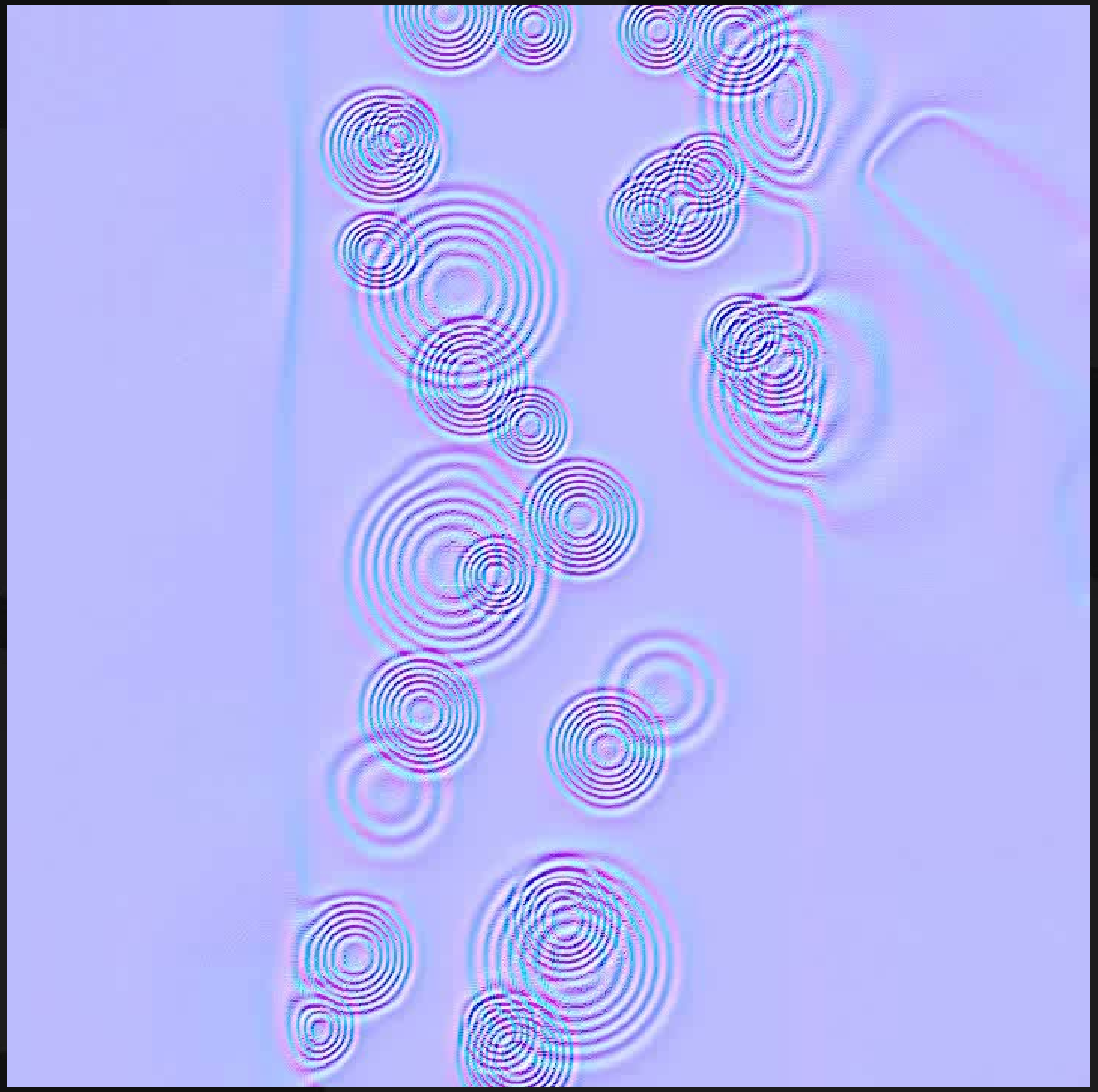
Very similar to emboss filter for U and V

- (Original Texture – Offsetting Texture) + 0.5 ≒ Emboss Filter

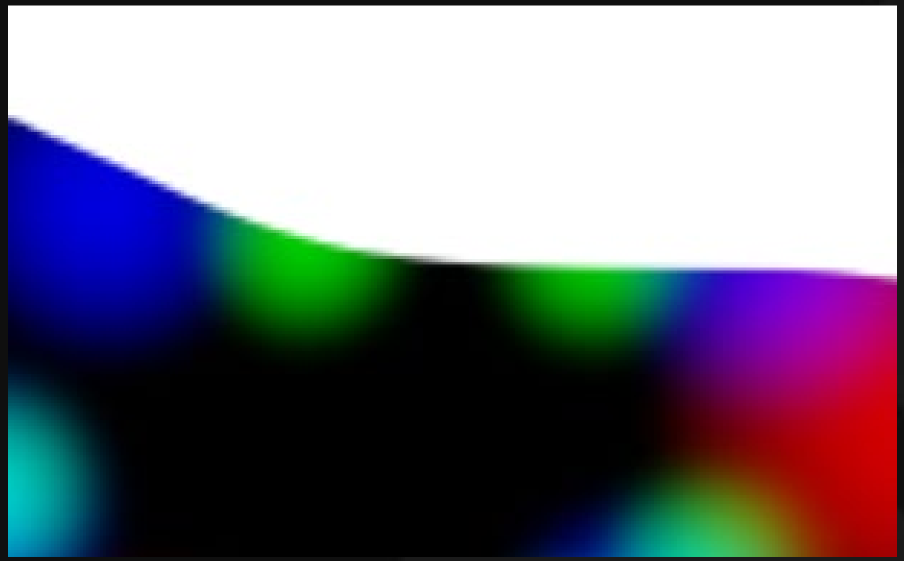


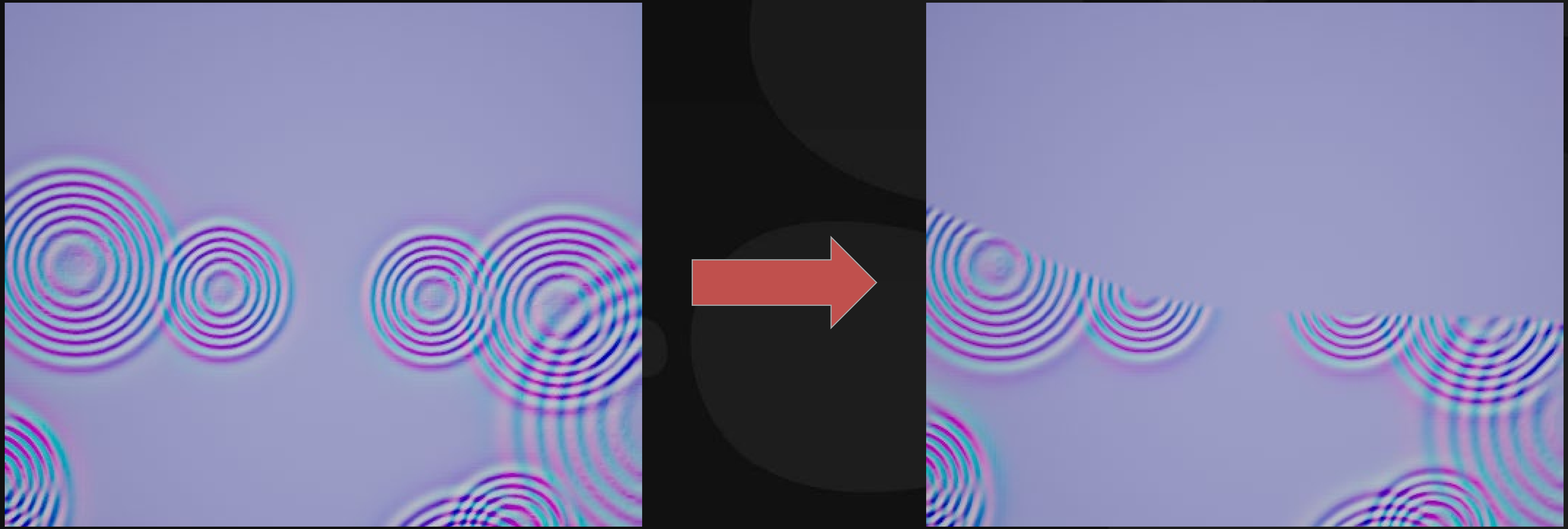
e.g. A texture covered 150 Meters

(1024*1024 RGBA-BC7 or 256*256 Uncompressed)



For 4 channels are Okay, if you want. But we used alpha channel as a mask map.

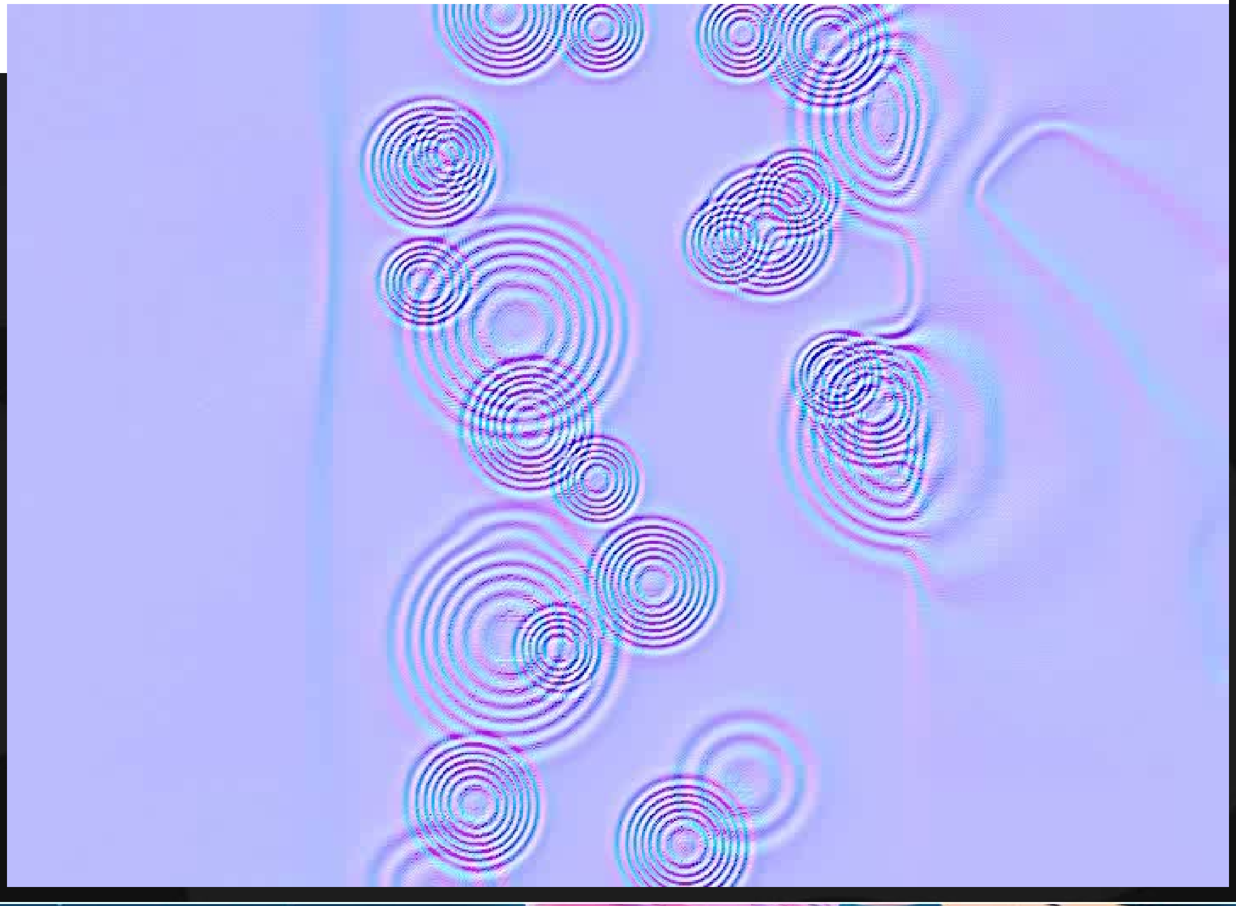




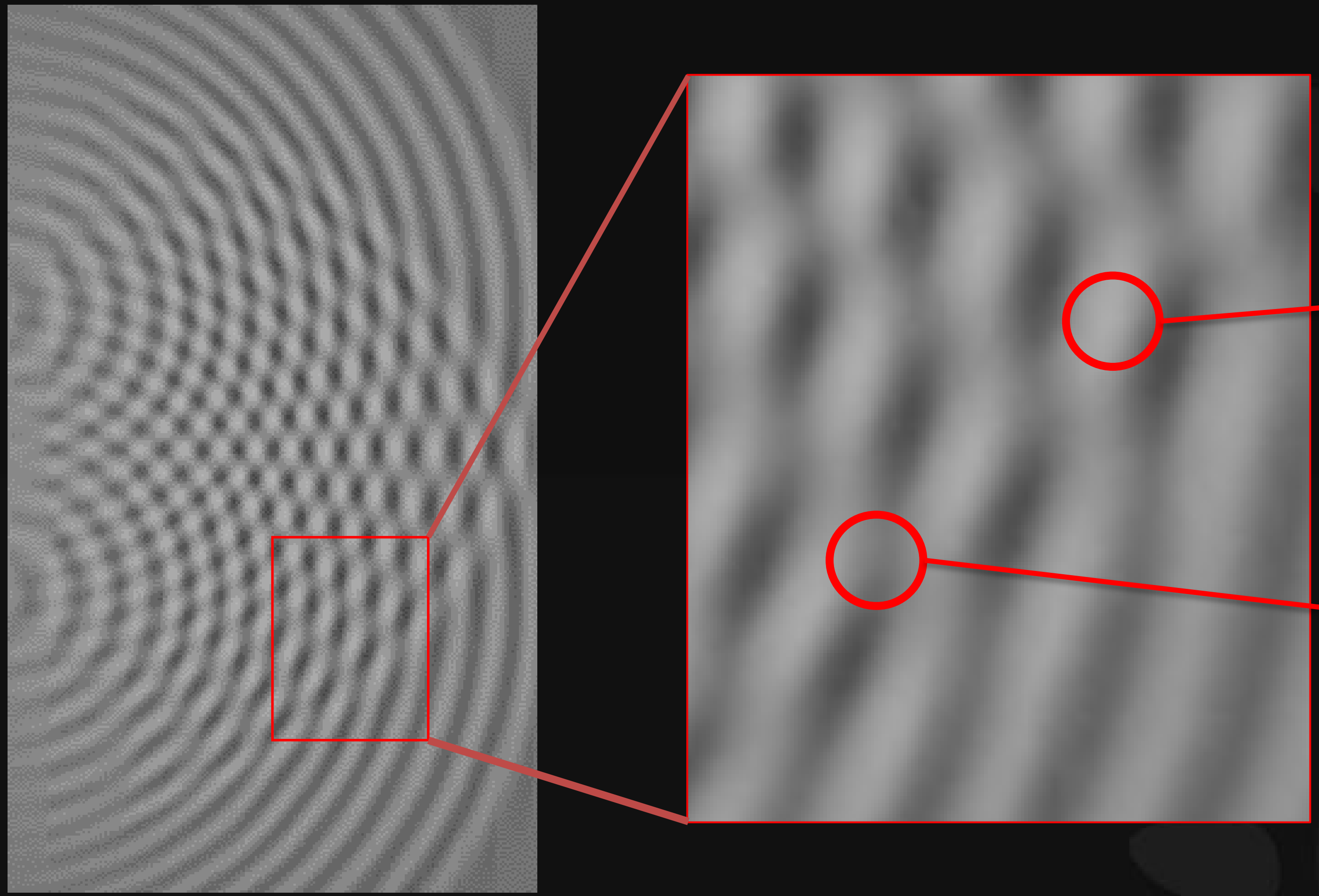
Like this…

```
float4 Original = tex2D(Sampler, UV + g_UVOfs * -0.72);
float4 OffsetX  = tex2D(Sampler, UV + float2(g_UVOfs.x, 0));
float4 OffsetY  = tex2D(Sampler, UV + float2(0, g_UVOfs.y));

float3 PreNmlU = (Original.xyz - OffsetX.xyz) * g_NormalStrength * Original.w;
float3 PreNmlV = (Original.xyz - OffsetY.xyz) * g_NormalStrength * Original.w;
float3 Normals = float3((PreNmlU.x + PreNmlU.y + PreNmlU.z + 0.5),
                        (PreNmlV.x + PreNmlV.y + PreNmlV.z + 0.5),
                        1.0);
```

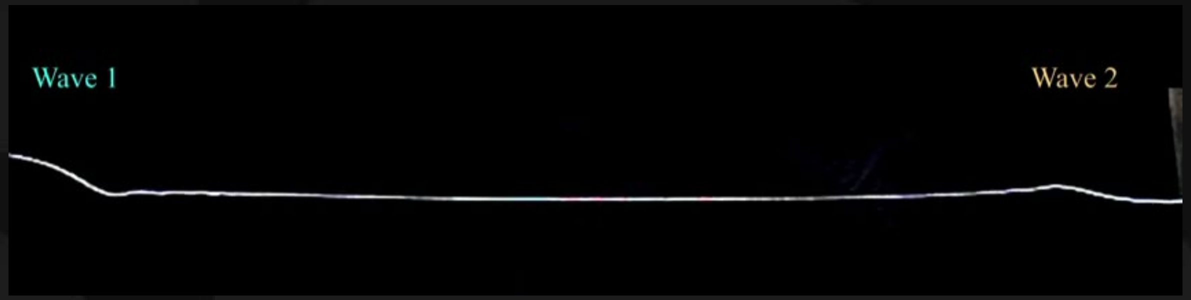


Constructive wave and Destructive wave are automatically reproduced



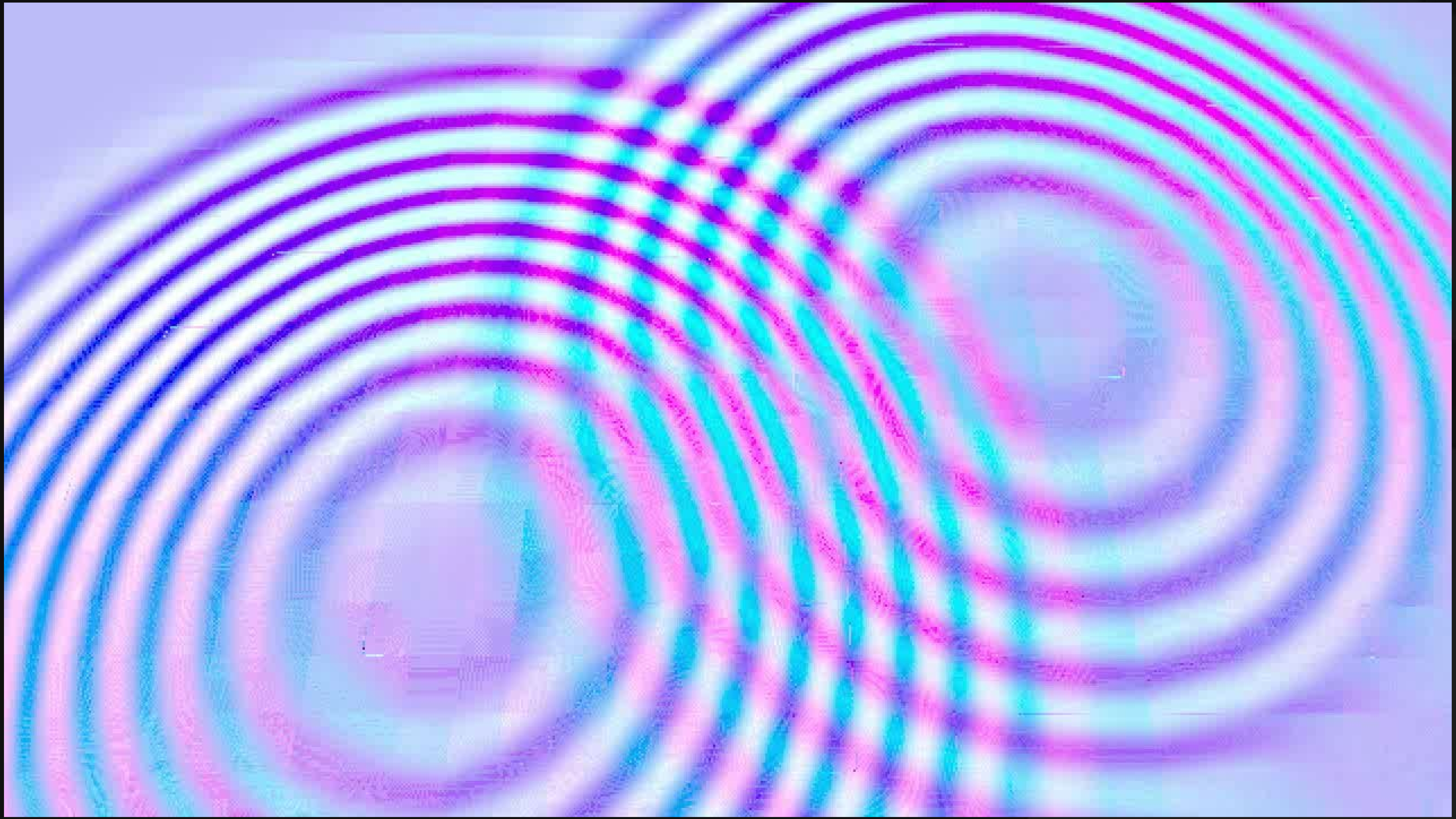
Two Source Interference
using our method

Constructive Wave; Wave1 + Wave2 = Double



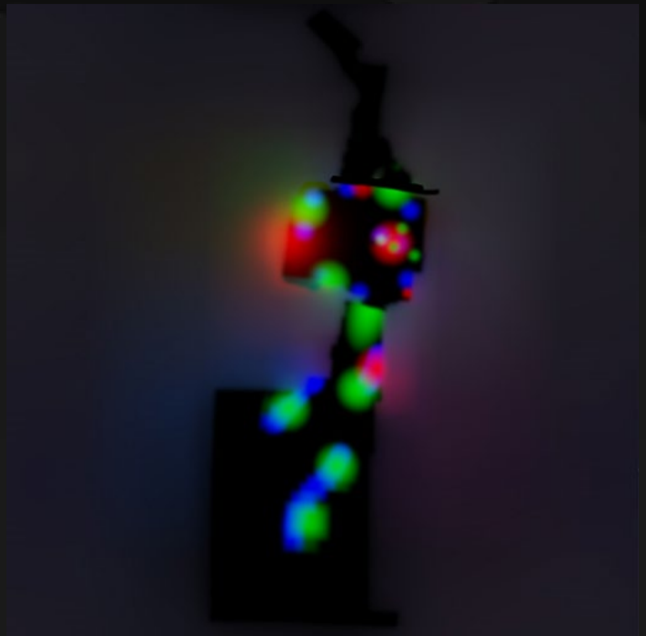
Destructive Wave; Wave1 + Wave2 = Zero

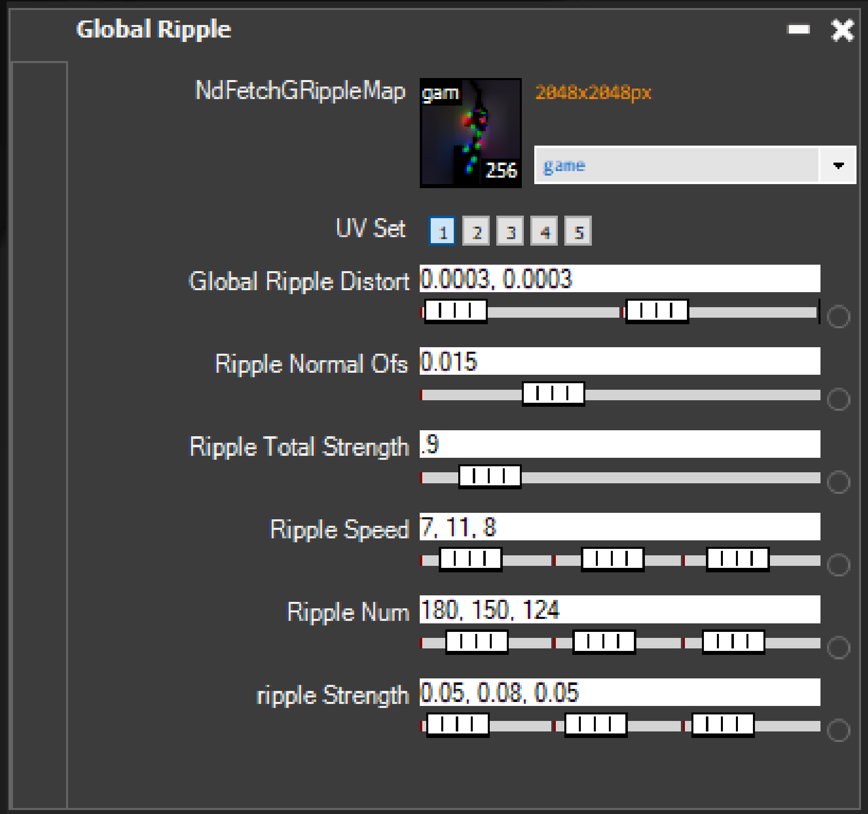
Comparison; DDX&DDY vs Our method



Comparison; the worst case on DDX&DDY vs Our method

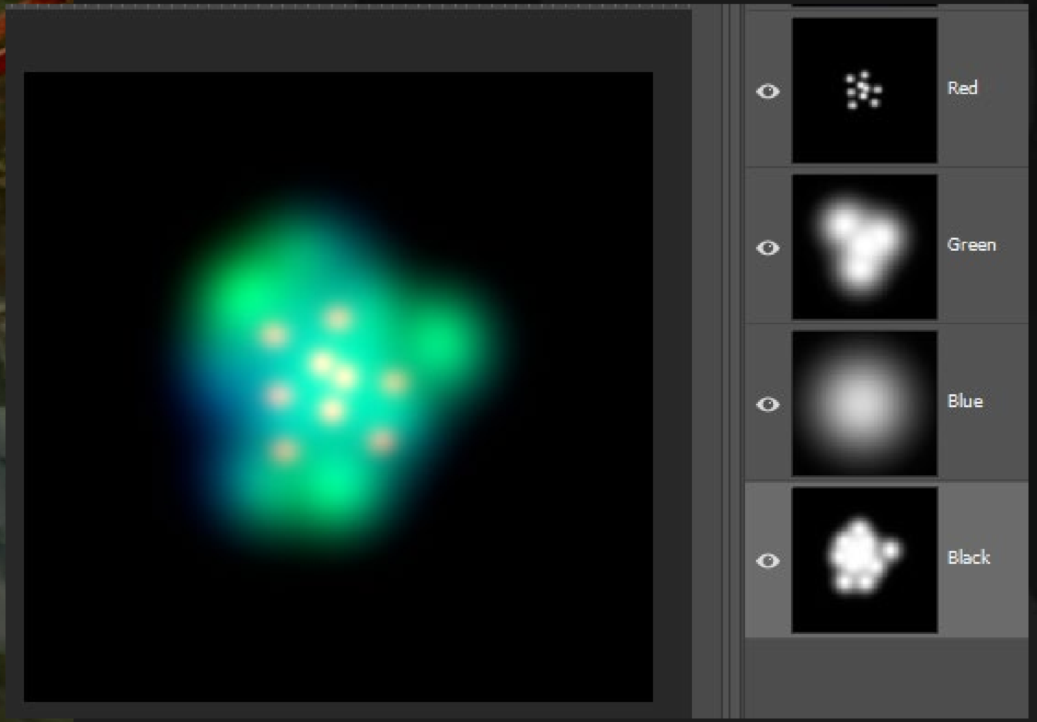
Parameters




Global Ripple
NdFetchGRippleMap
gam
256
2048x2048px
game
UV Set
1 2 3 4 5
Global Ripple Distort
0.0003, 0.0003
Ripple Normal Ofs
0.015
Ripple Total Strength
.9
Ripple Speed
7, 11, 8
Ripple Num
180, 150, 124
ripple Strength
0.05, 0.08, 0.05

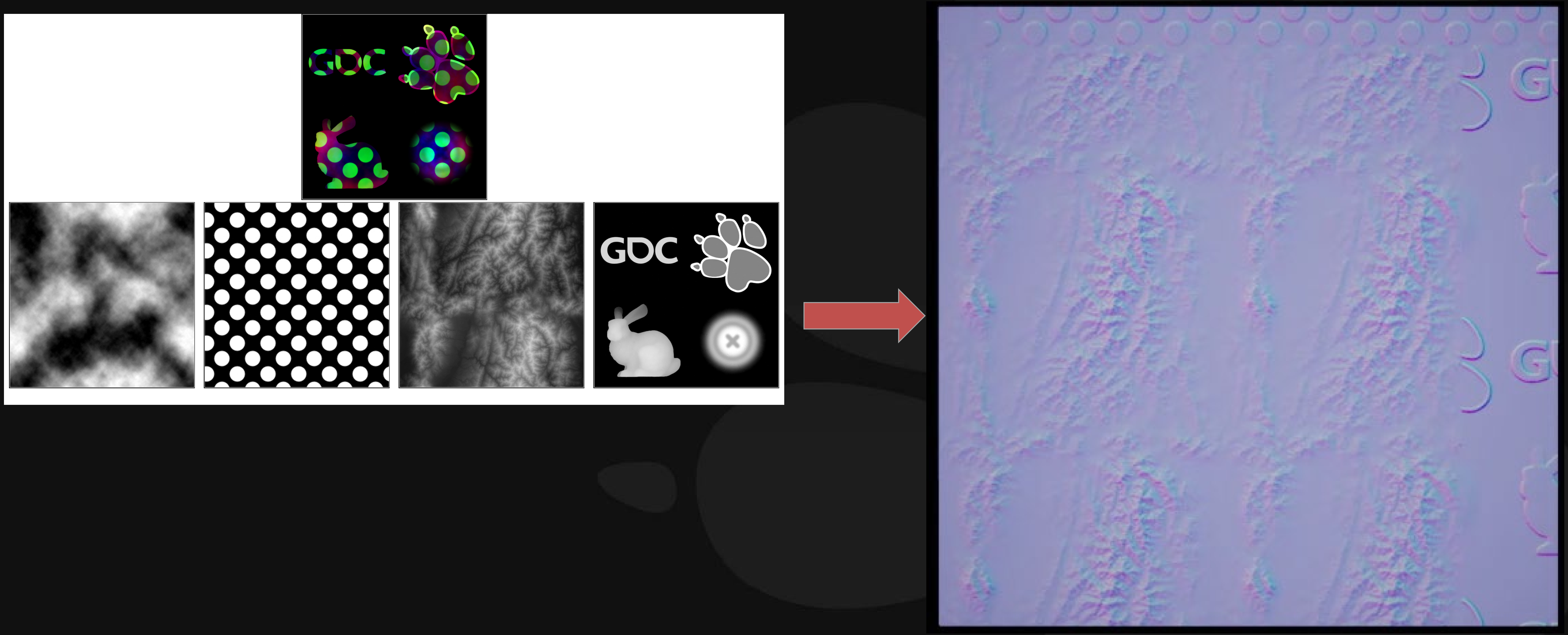
It works for particle system as well ☺

- Projected to water surface as normal map.



As Proof

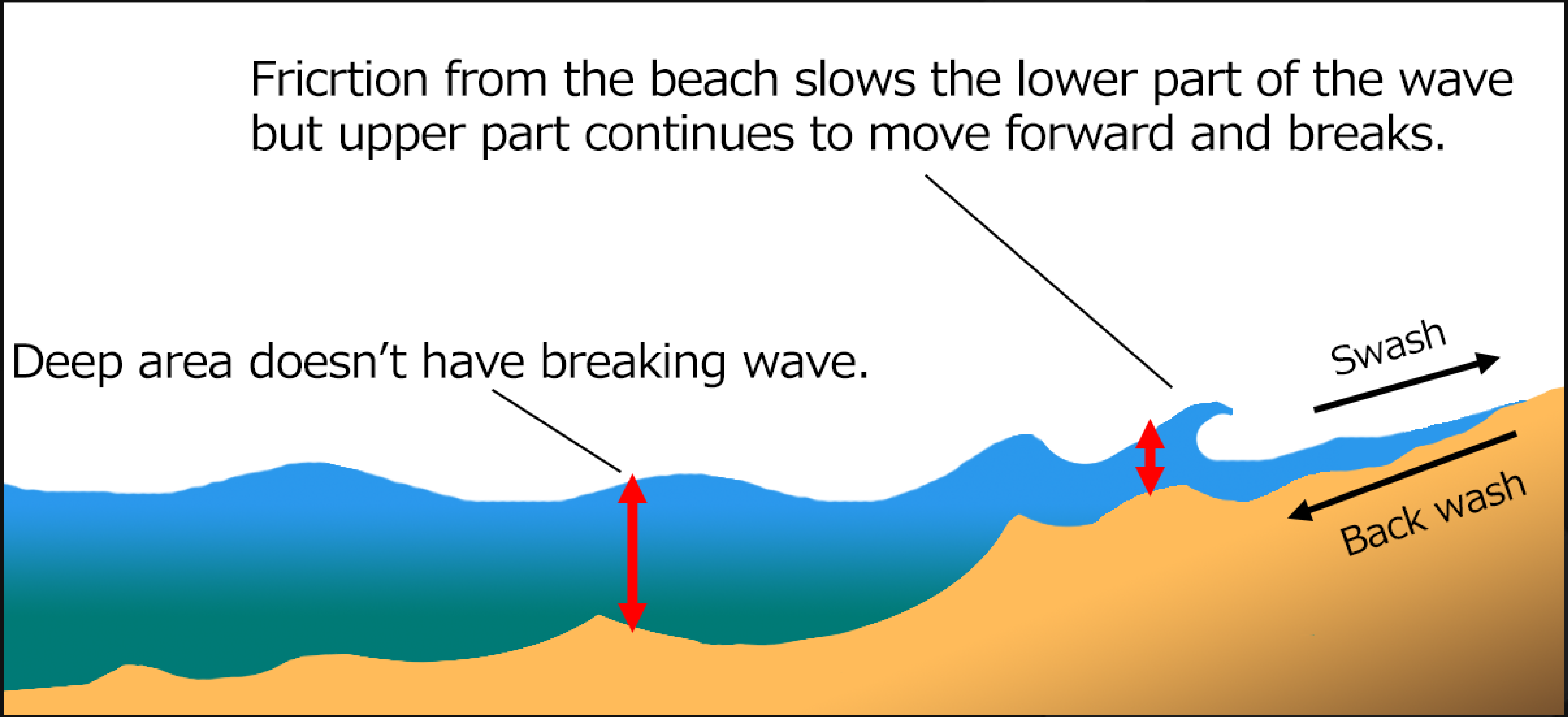
4 different generated normal maps from a texture



CURL WAVE AT BEACH

Wave? Naughty Dog has own ocean wave system already

Real Wave Action



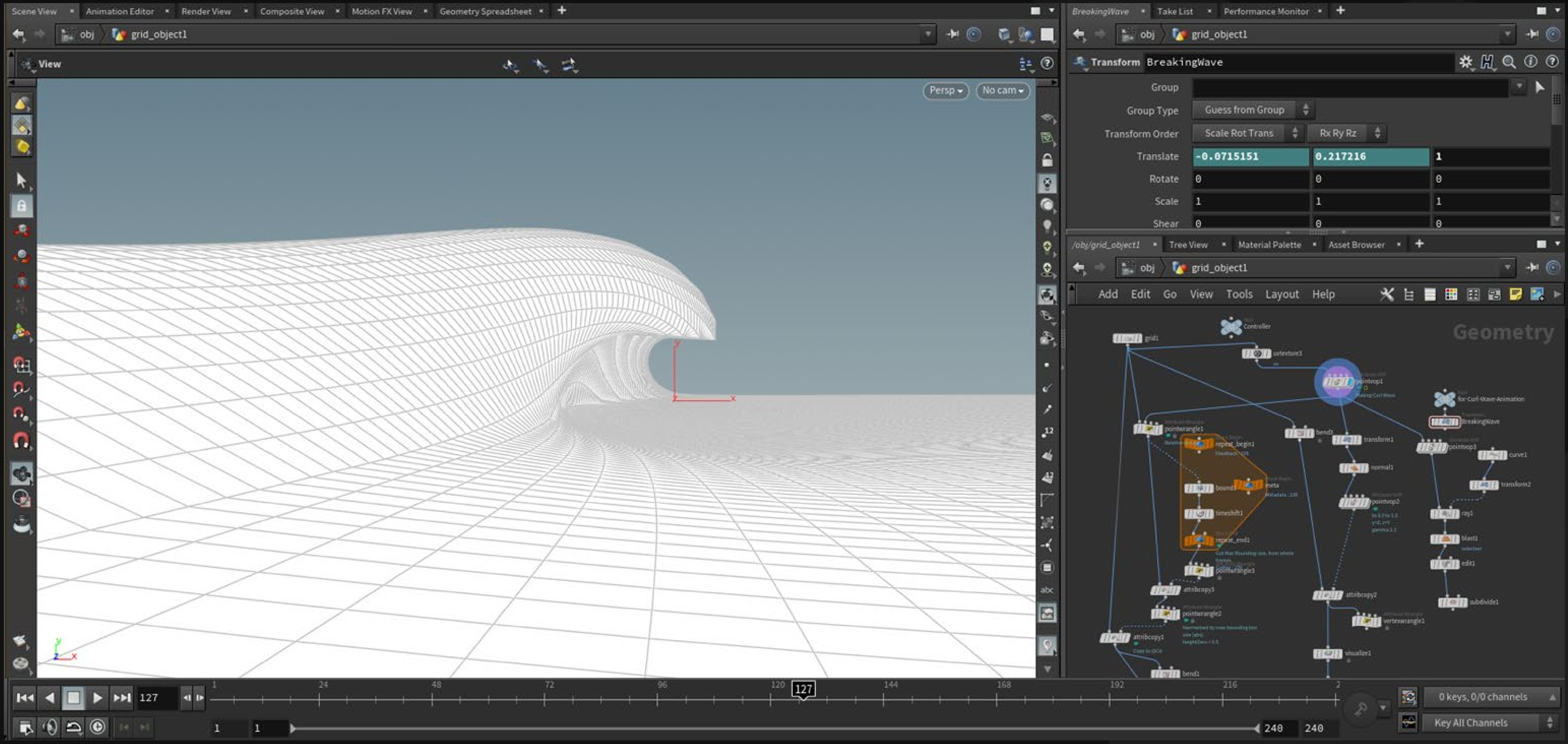
Existing available G-buffers for Ocean System

Vertex Offset

R=Foam, G=Churn, B=Algae

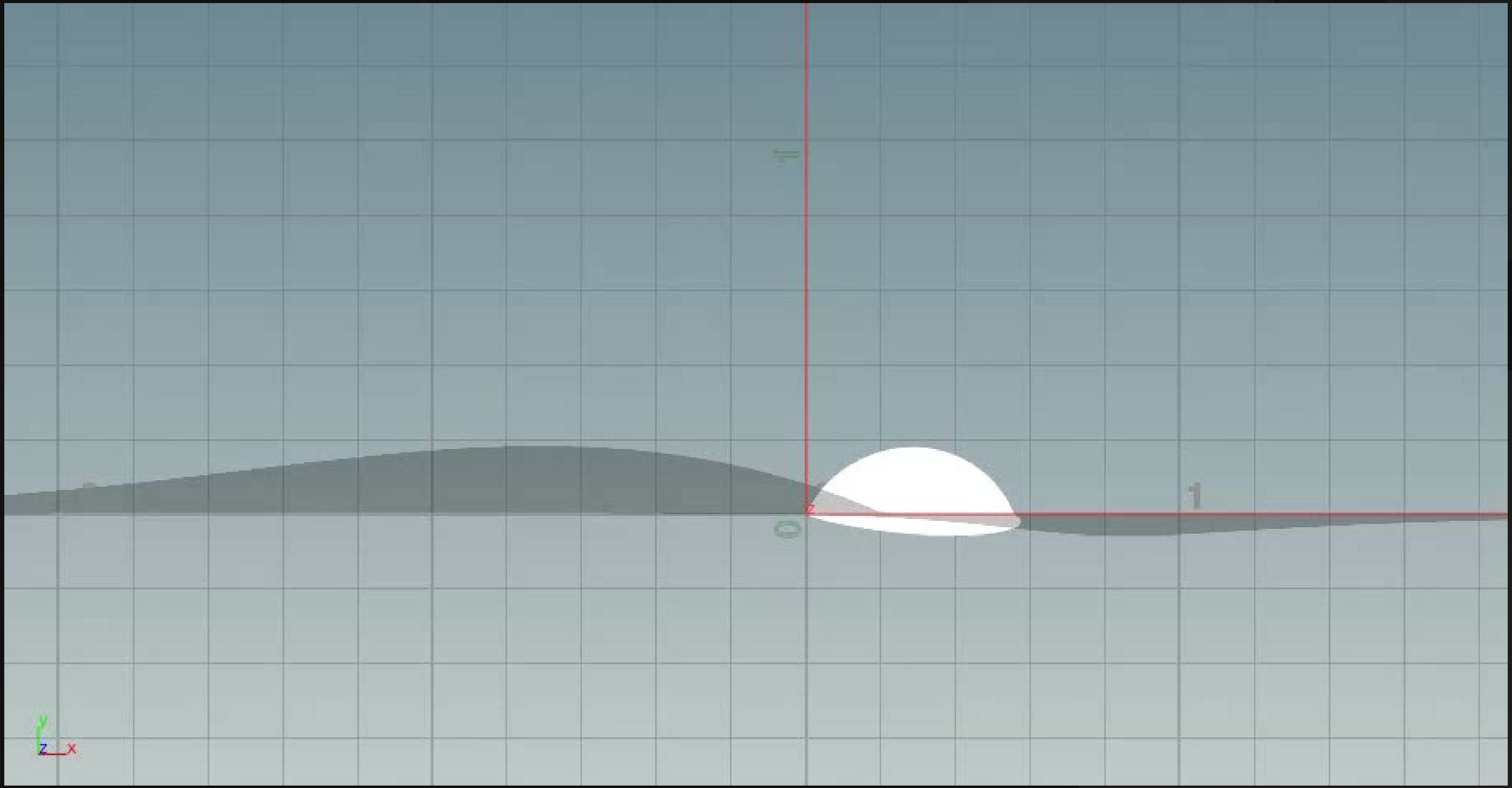
Normal

Making Curl Wave (Breaking Wave) with Houdini

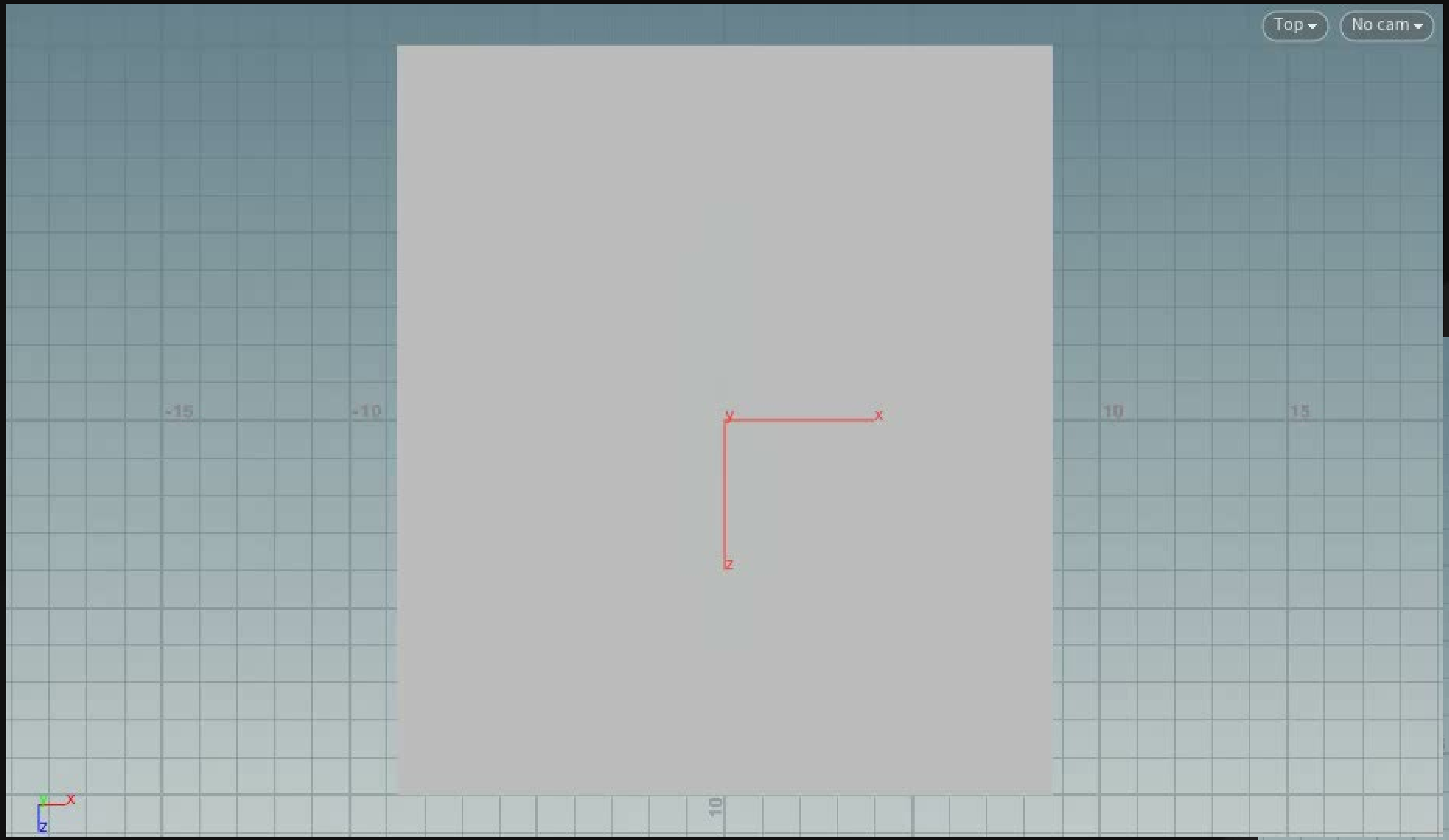


Making Displacement Map and Normal Map with Houdini

Normalized Wave form in order to bake to texture

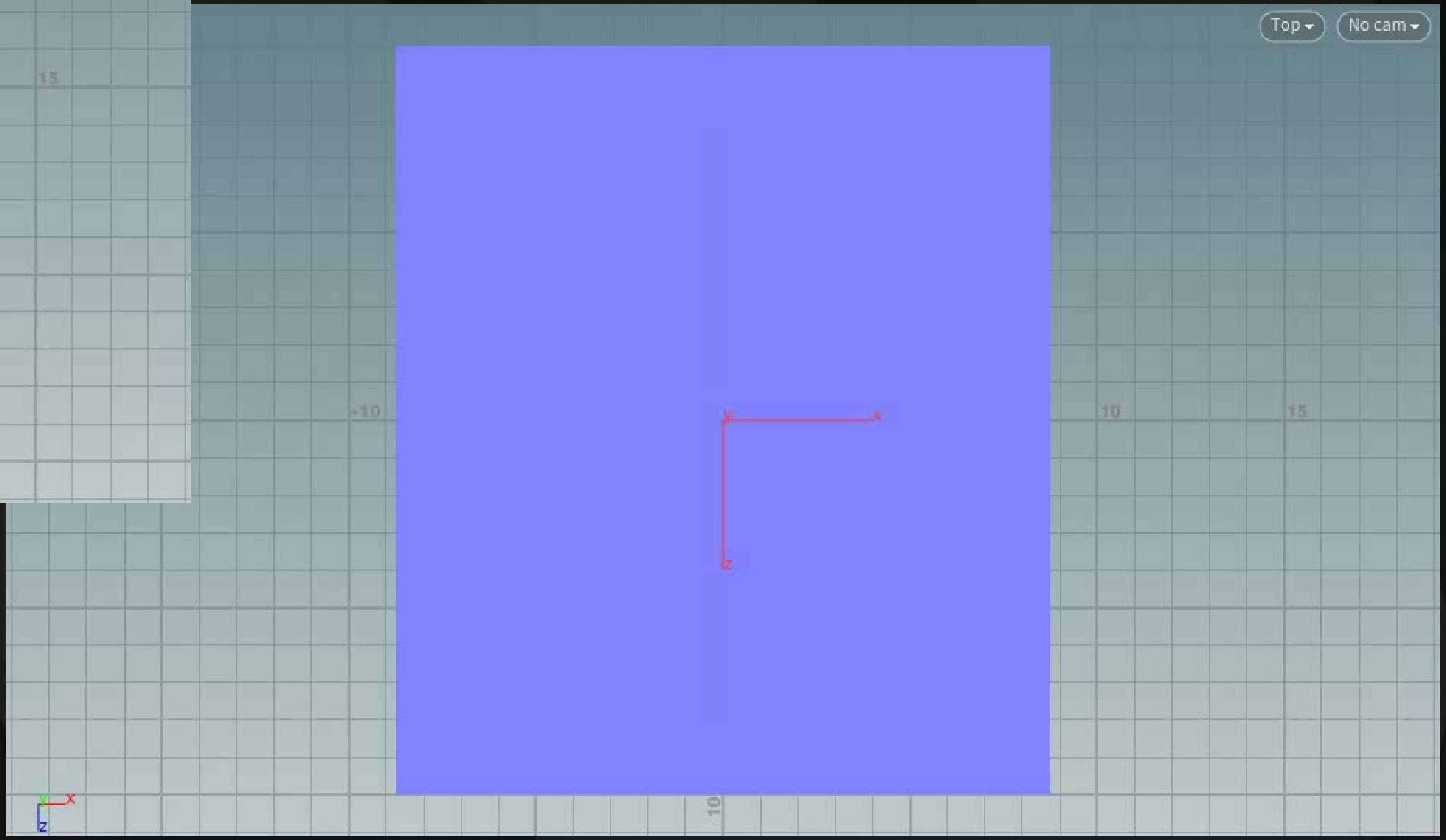


Baking to textures

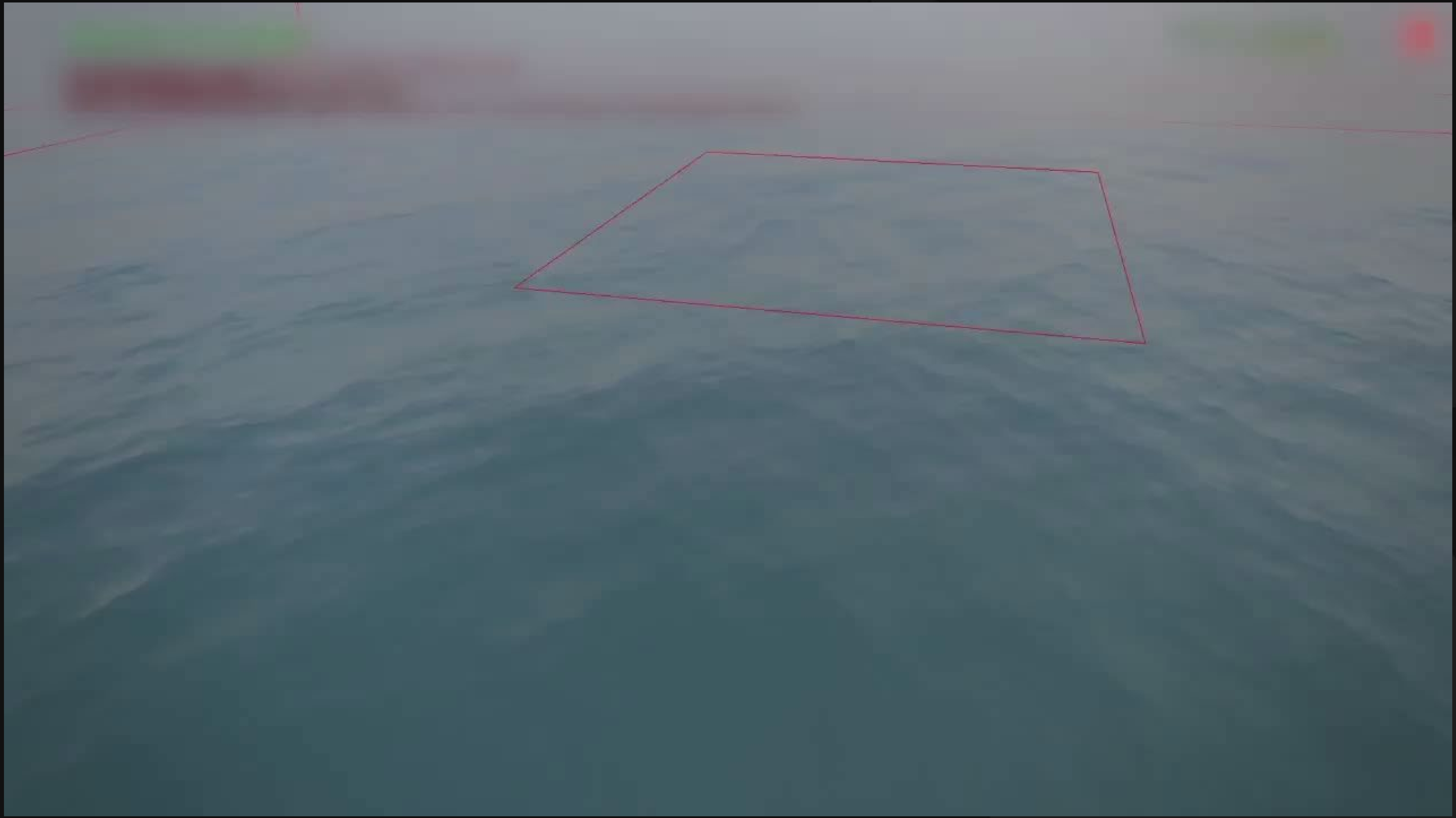


Displacement Map

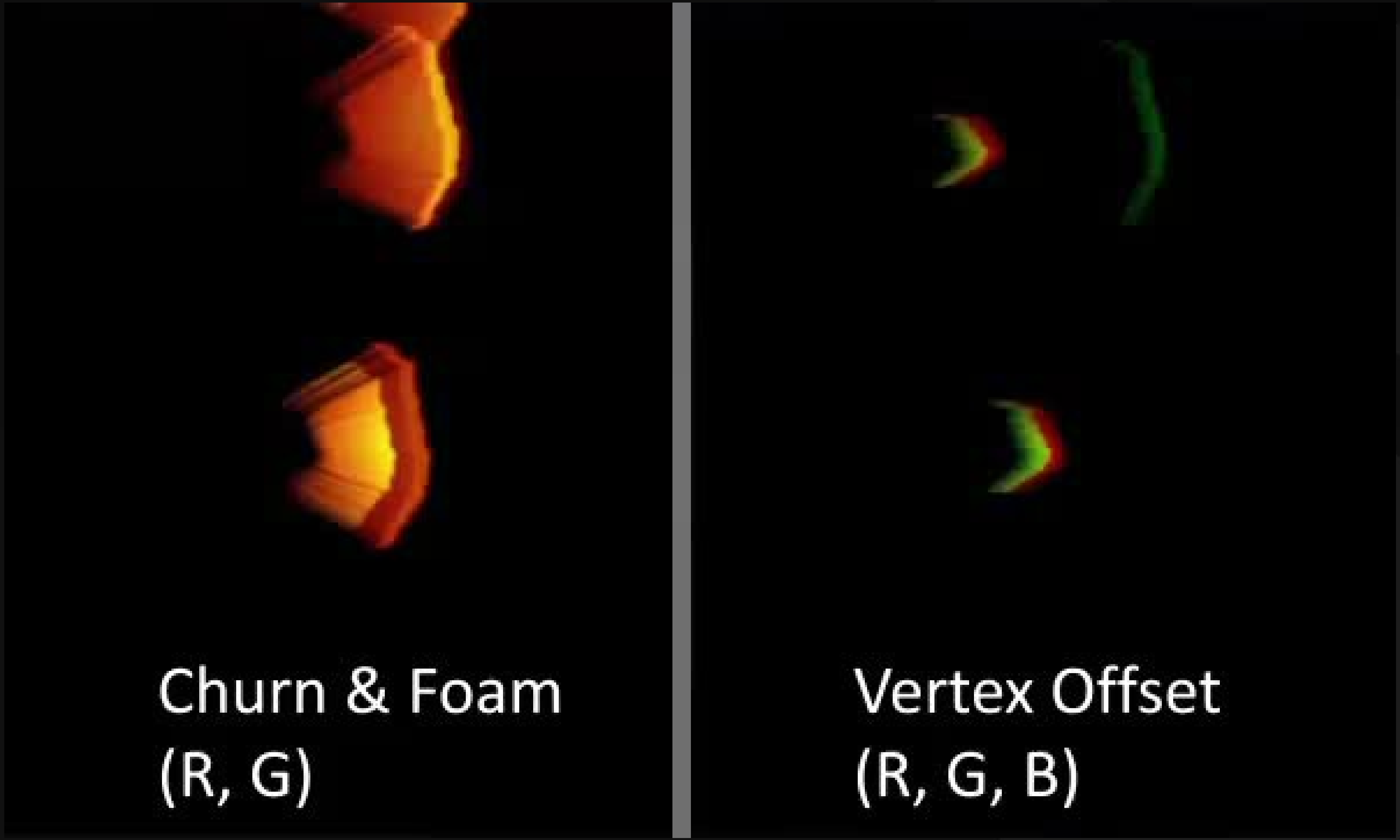
Normal Map



Projected Displacement Map to ocean



Projected Displacement, Churn and Foam are using g-buffer

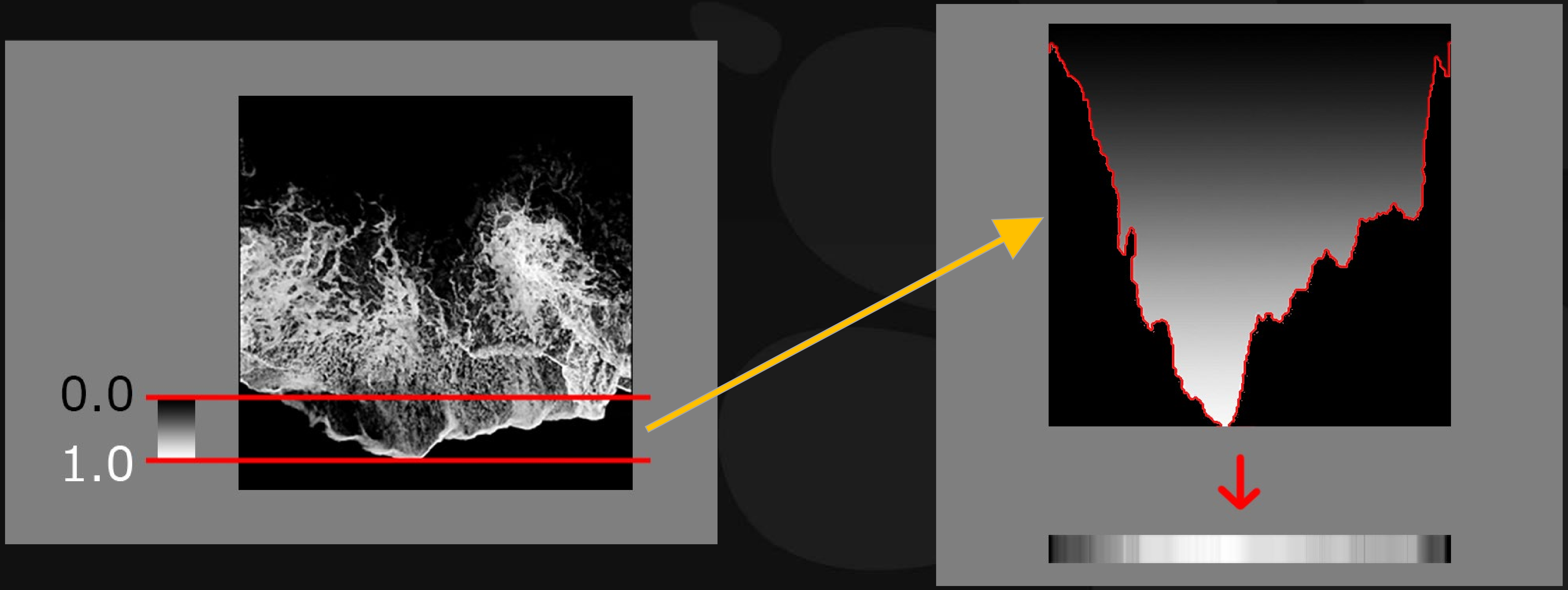


The foam of the waves had a triangular shape

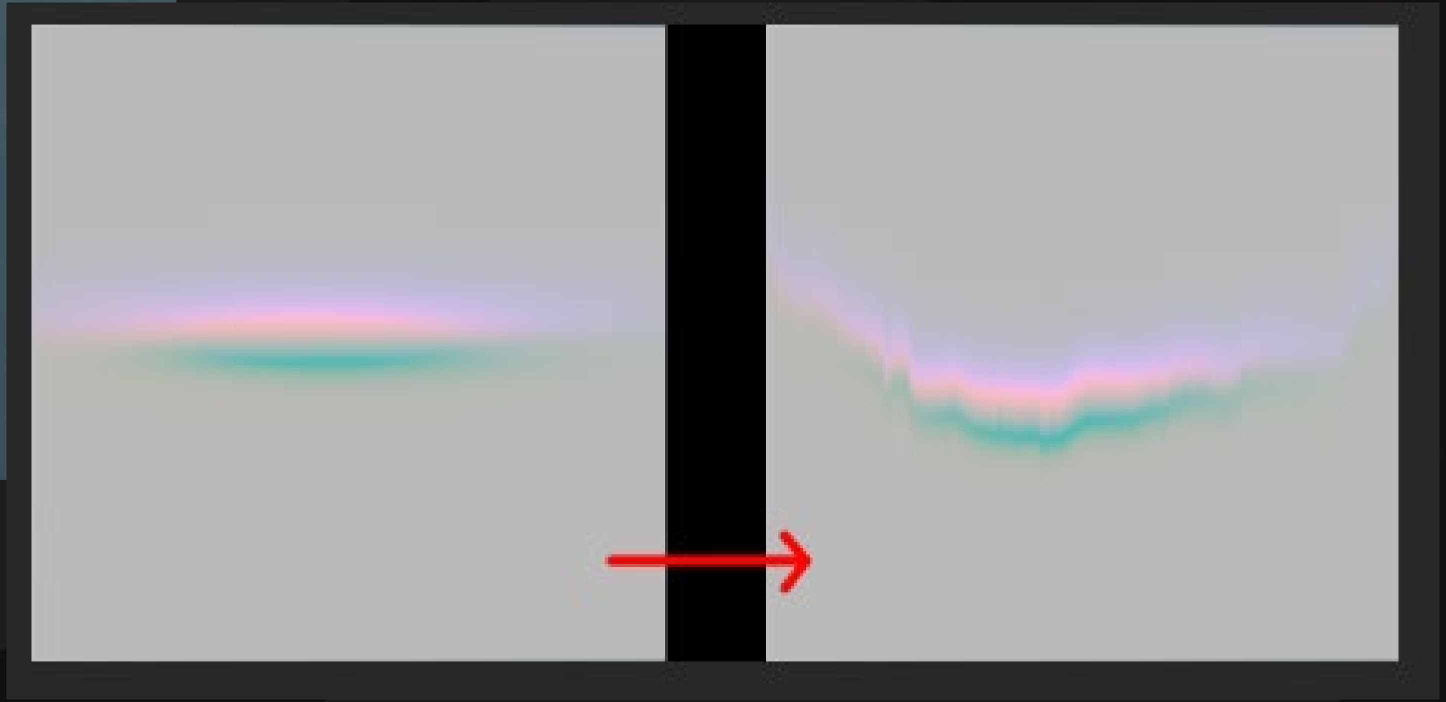
Reference: Mixkit

Experiment; Adjustment to projected foam's edge at beach

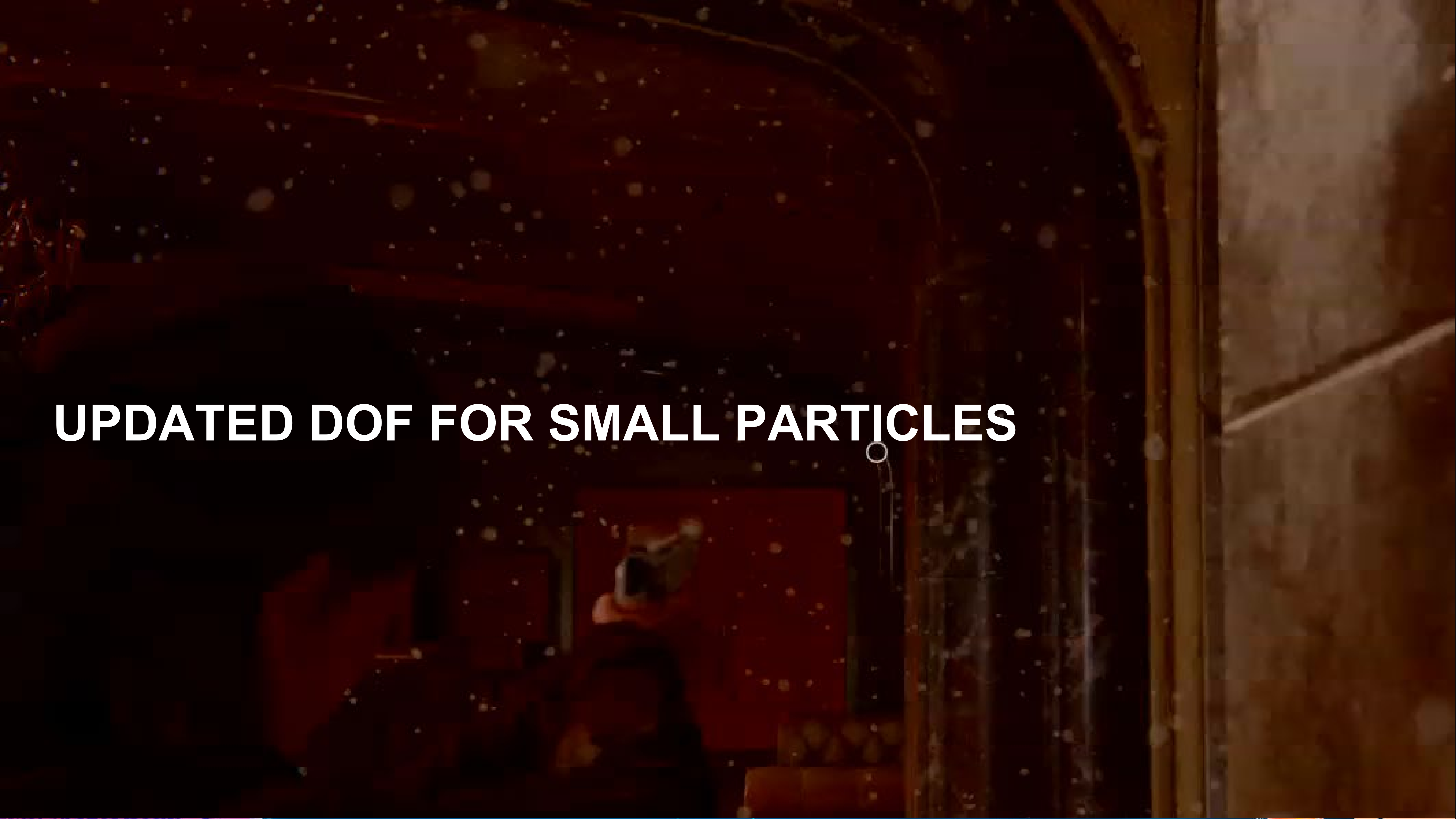
- It's simple. Getting the difference of head's outline of the wave.
- Normalize to 0.0~1.0. It will be distortion amount.



Adjustment to projected foam's edge at beach

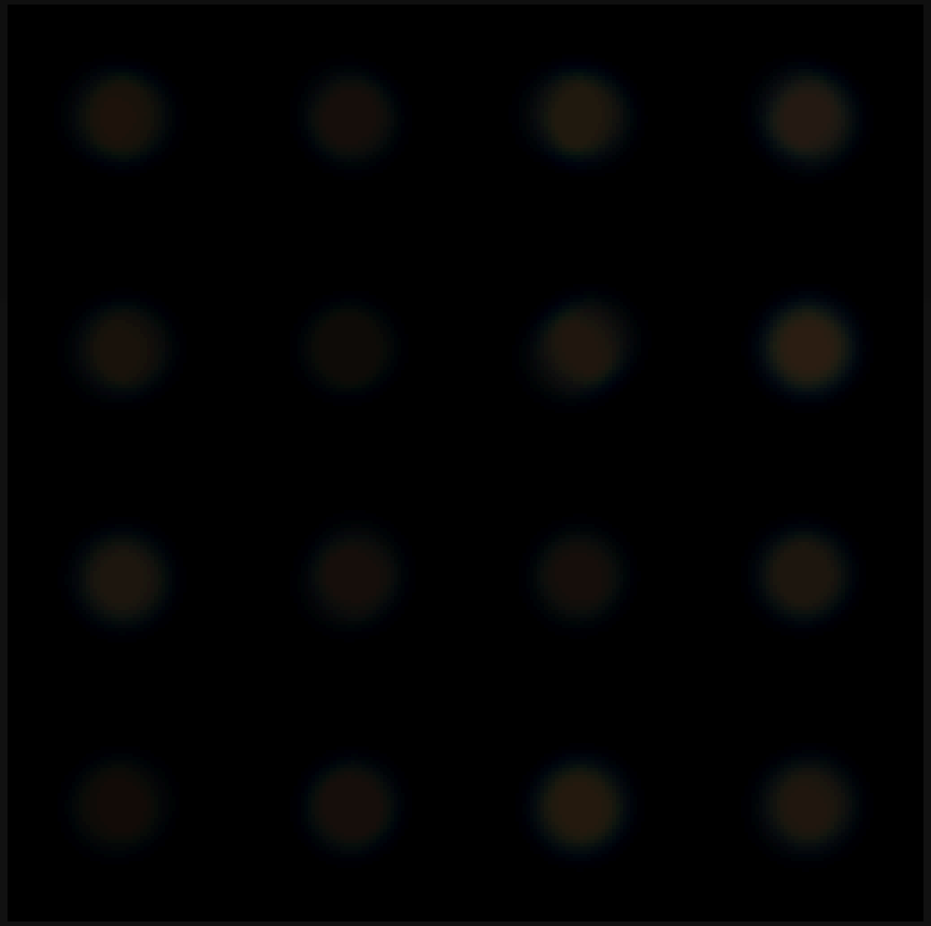


UPDATED DOF FOR SMALL PARTICLES



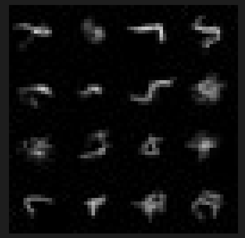
Our existing Texture DoF system

- 3 steps bokeh * 2 textures = 6 steps
- A step focused (no bokeh) texture = 1 step
- Total 7 steps using 3 textures



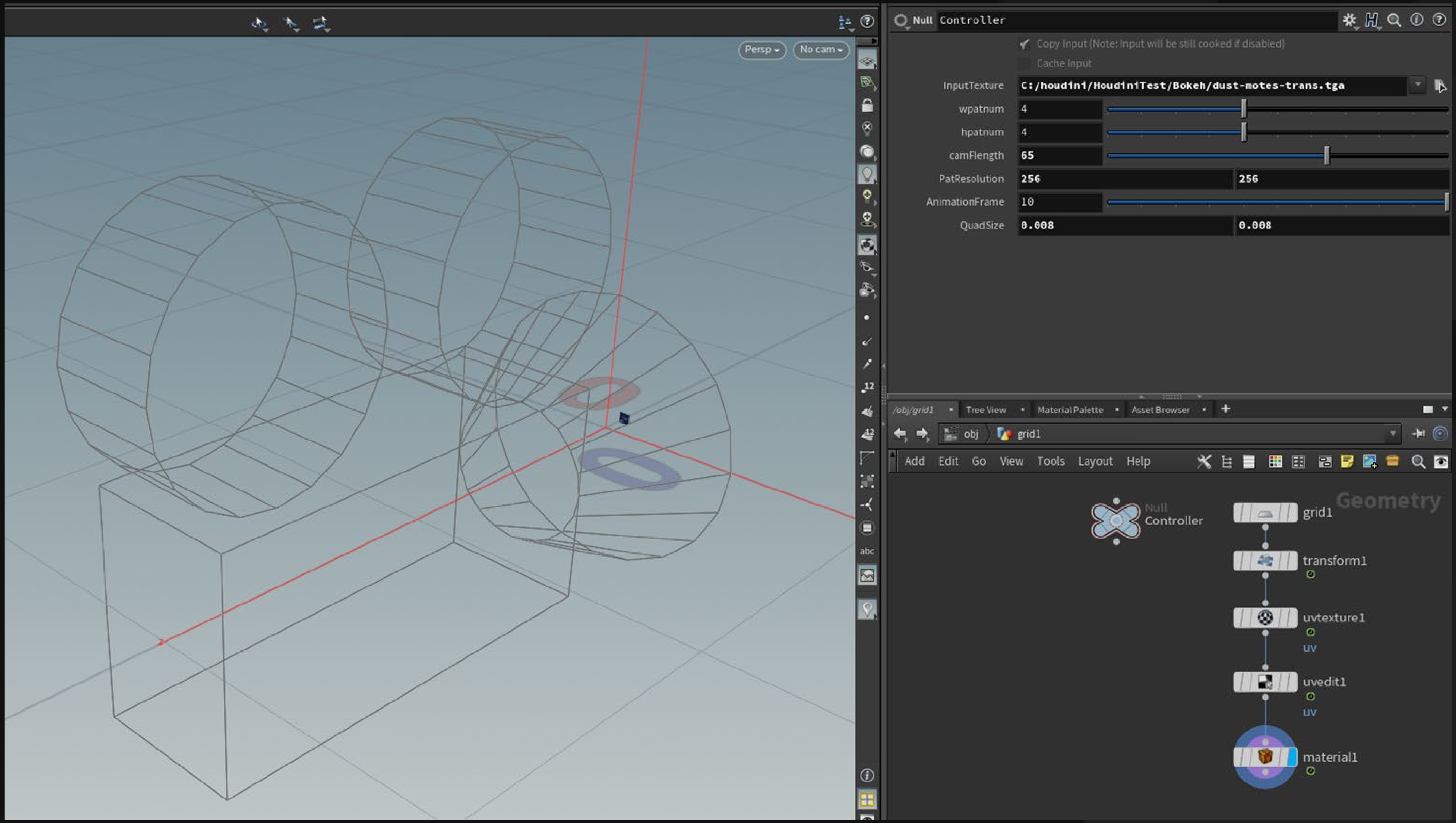
More Bokeh(R, G, B)

Bokeh (R, G, B)

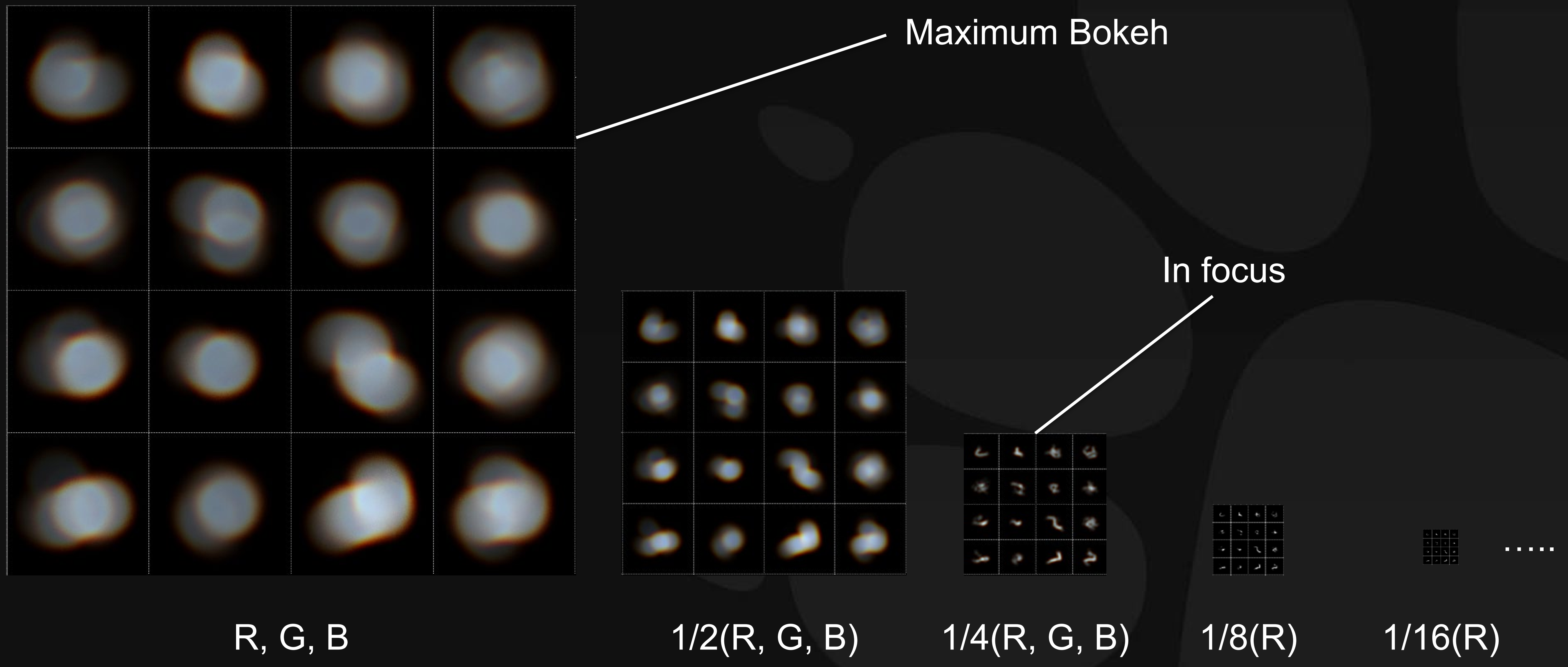


Focused (R)

Making Texture with Houdini

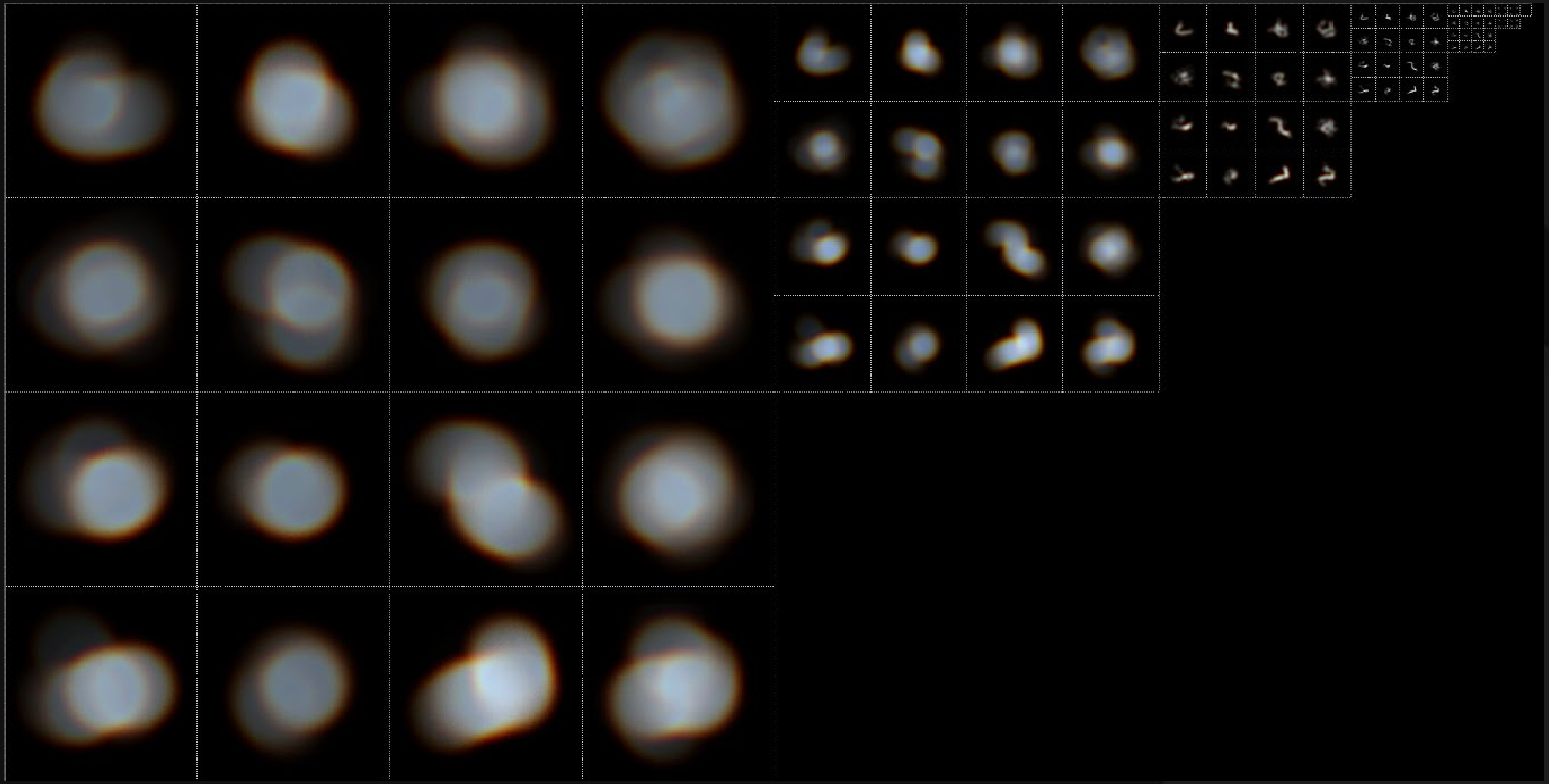


Rendered Images



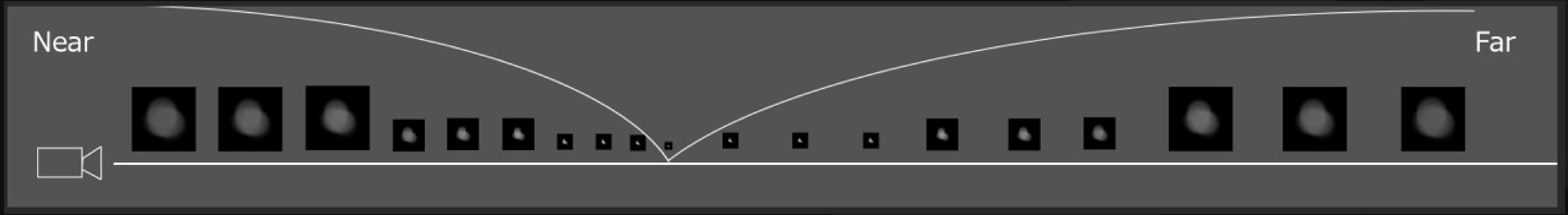
Optimization

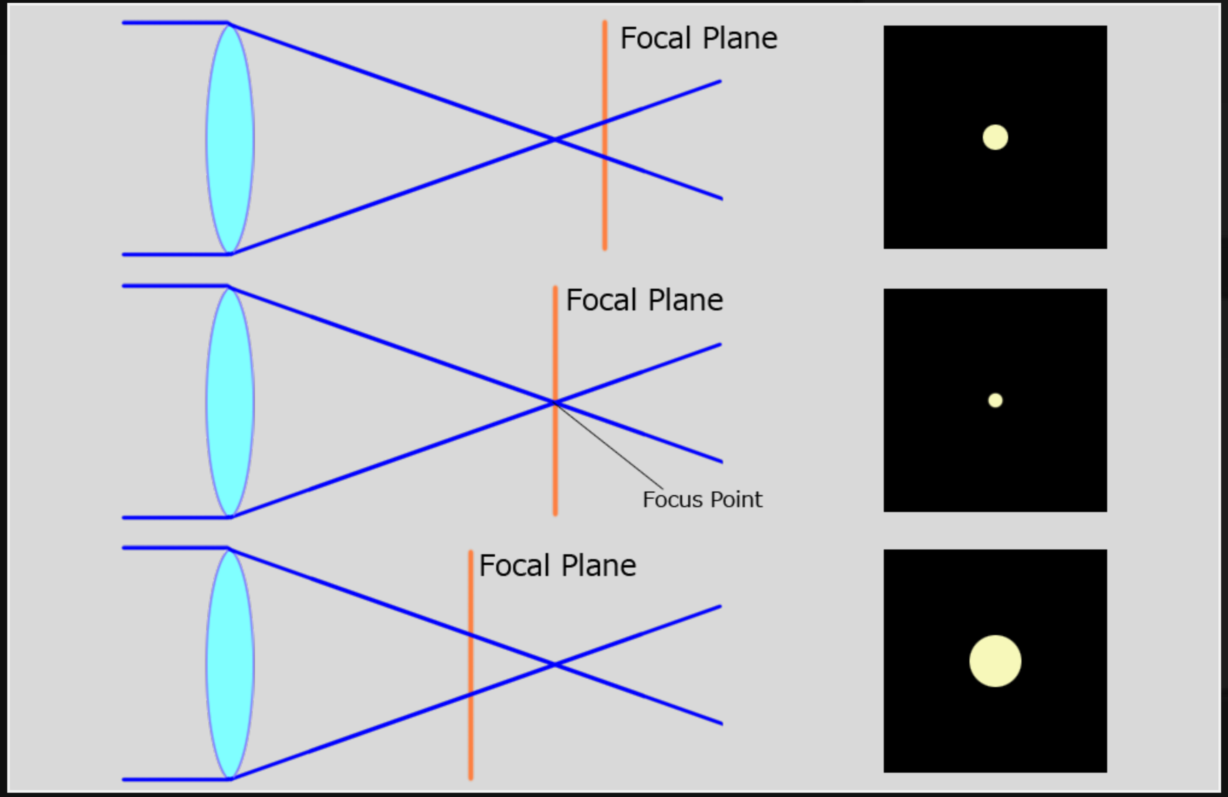
2~4 textures >> 1 texture using Mipmaps.



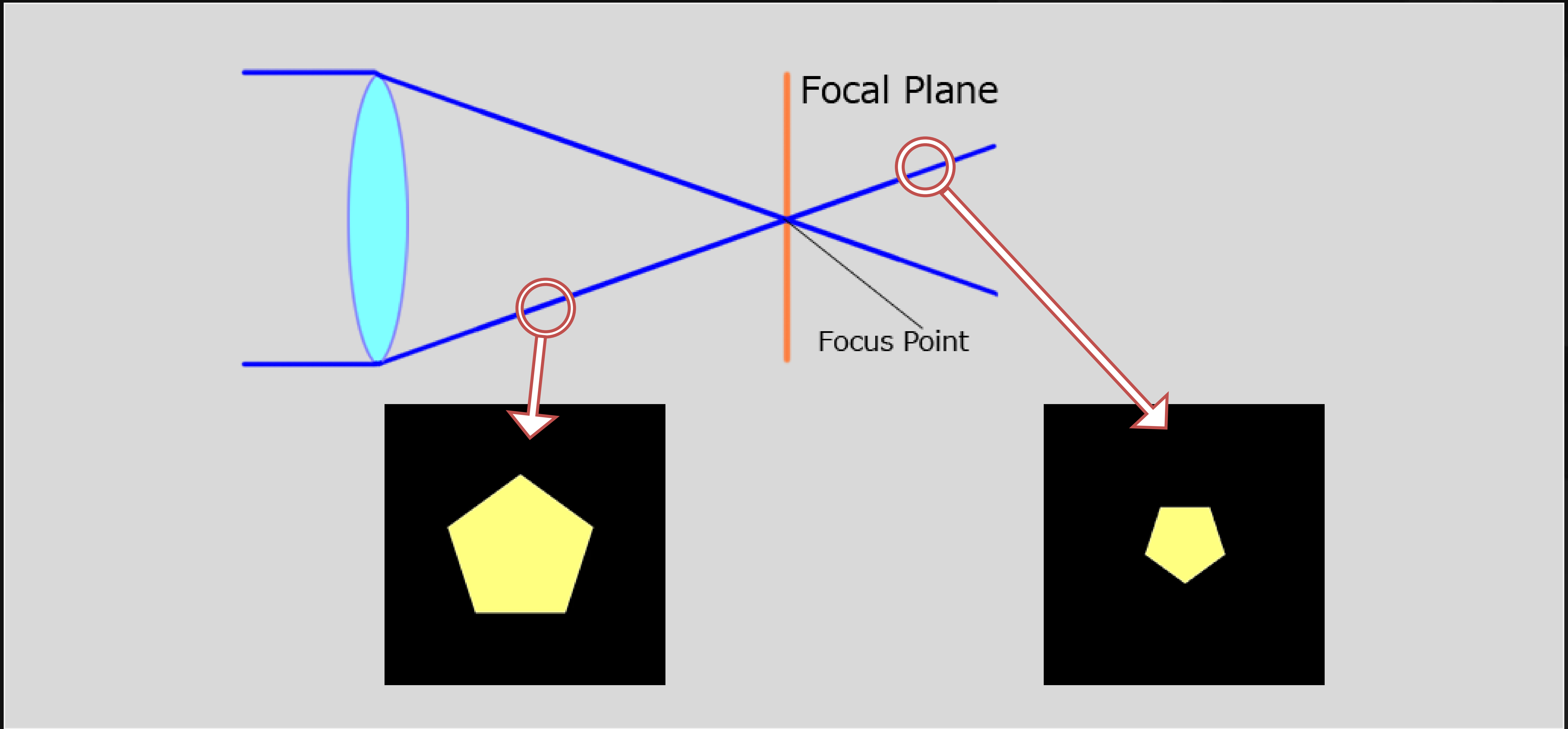
How does it work?

The Camera's CoC and the channel & MIP levels used are linked and transition appropriately.

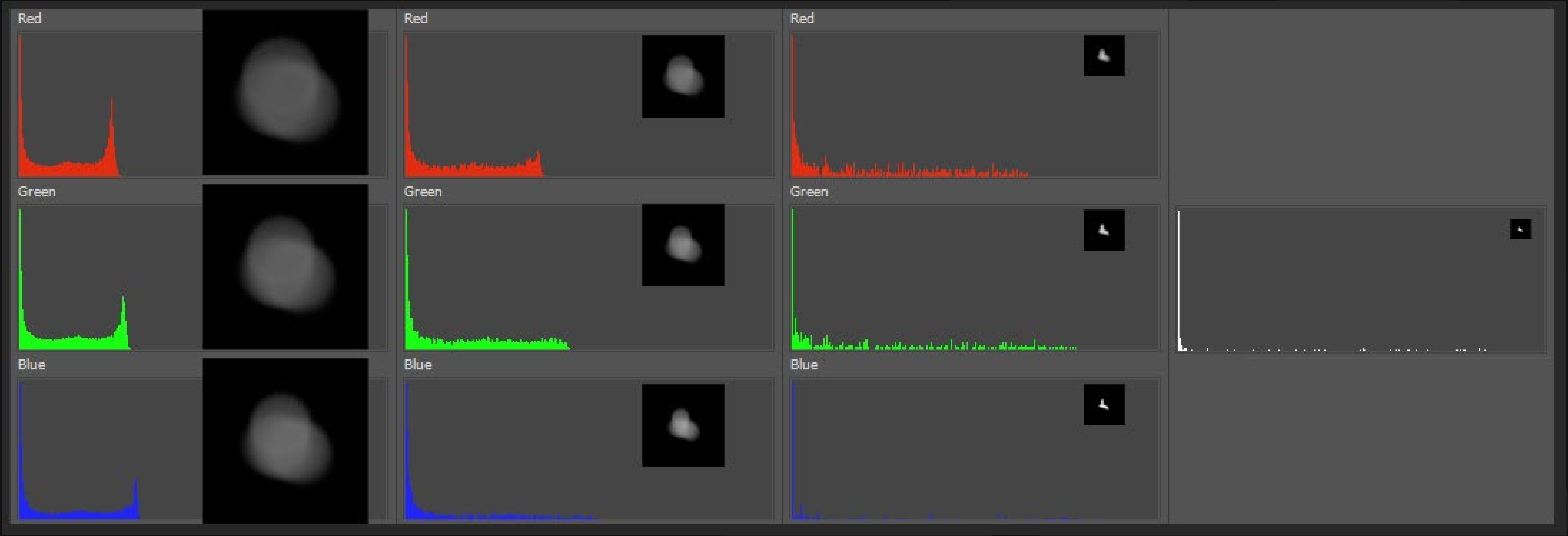




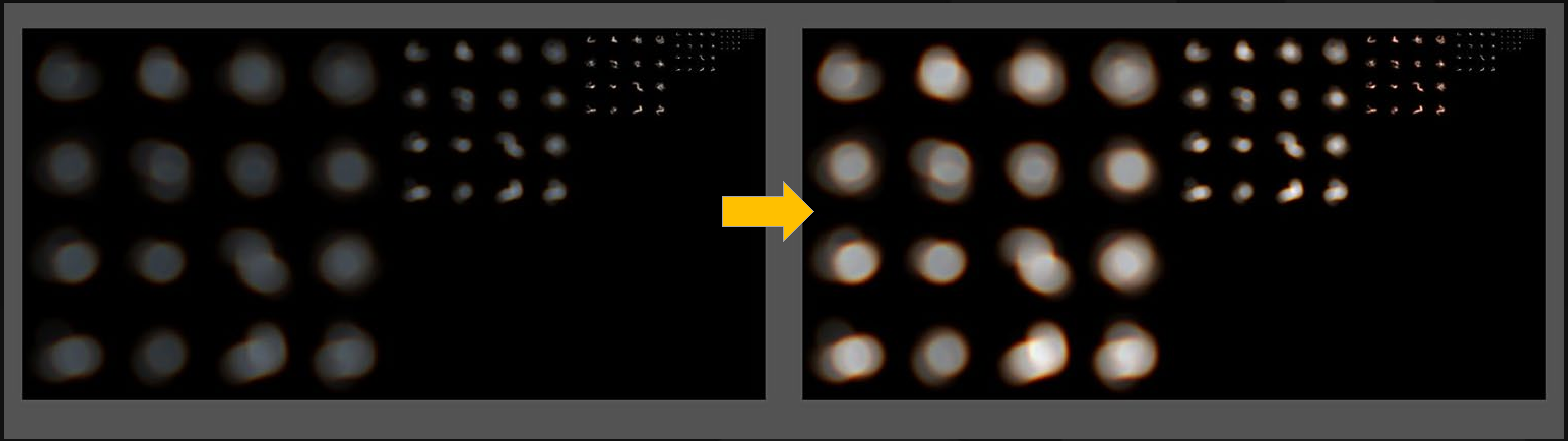
Tips



Bokeh was not using much tonal range…



Adjusted Translucency in advance



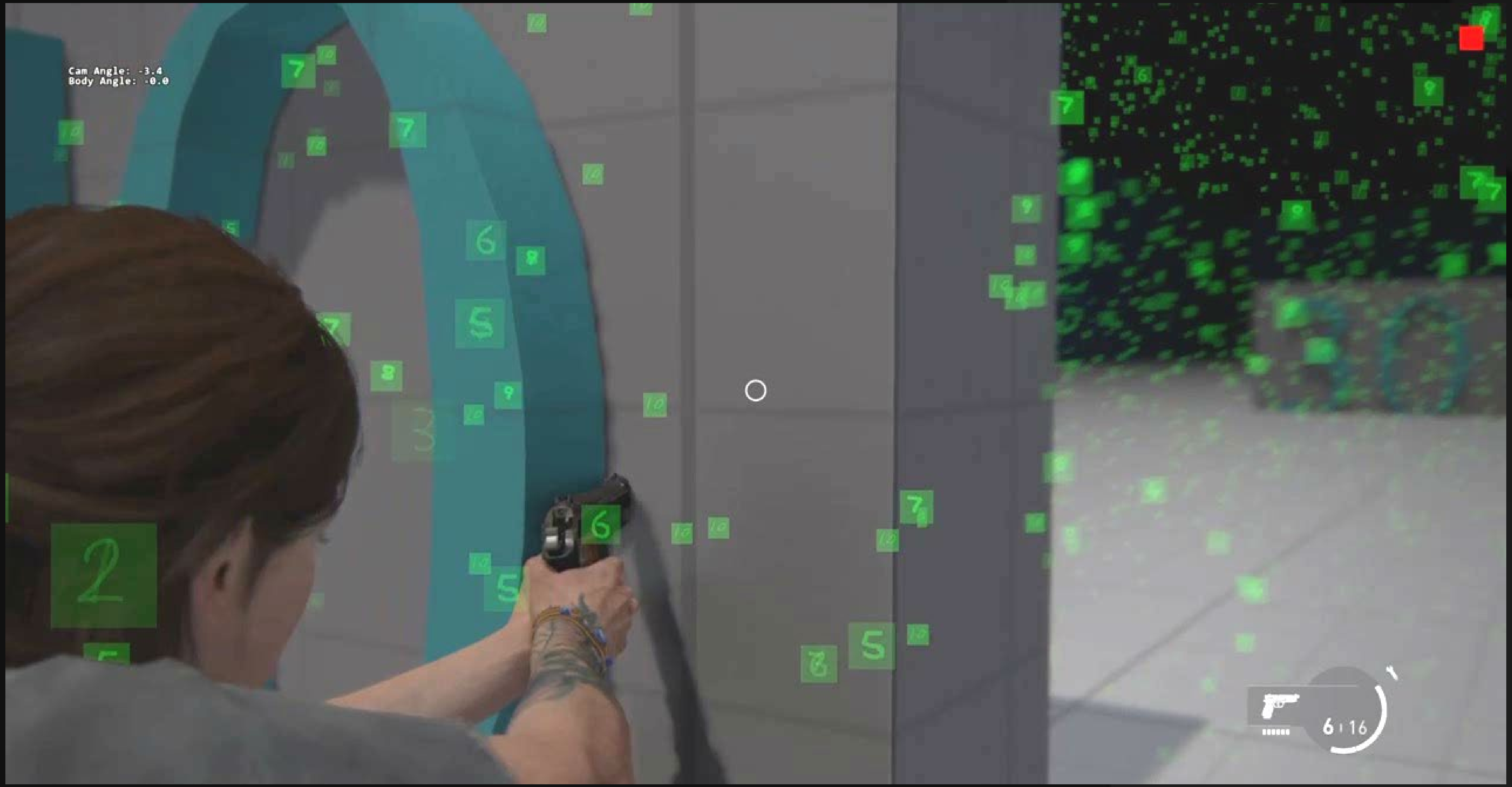
Before

After (using whole range)

And then it's adjusted to the original by shader.

Use cases

- For Dust-motes, Additional spores, Falling Snow and so on…
- In Focus shows (9). The most Bokeh(Out of Focus) shows (1).



Next Step

- Baking ripple sim to texture channel or figure out low-cost real-time sim.
- Procedural curl wave on the appropriate position automatically.
- Real-time simulation for Texture DoF somehow.

With artistic controllable of course!

Acknowledgements

All VFX Team Members;
In particular, Eben Cook, Raymond Popka, Quinn Kazamaki and Elaine Kubik

Programming Team;
In particular, Artem Kovalovs

Also Environment Team, Character Team
and all project personnel of The Last of Us Part II

Naughty Dog is hiring!

https://www.naughtydog.com/careers

NAUGHTY DOG