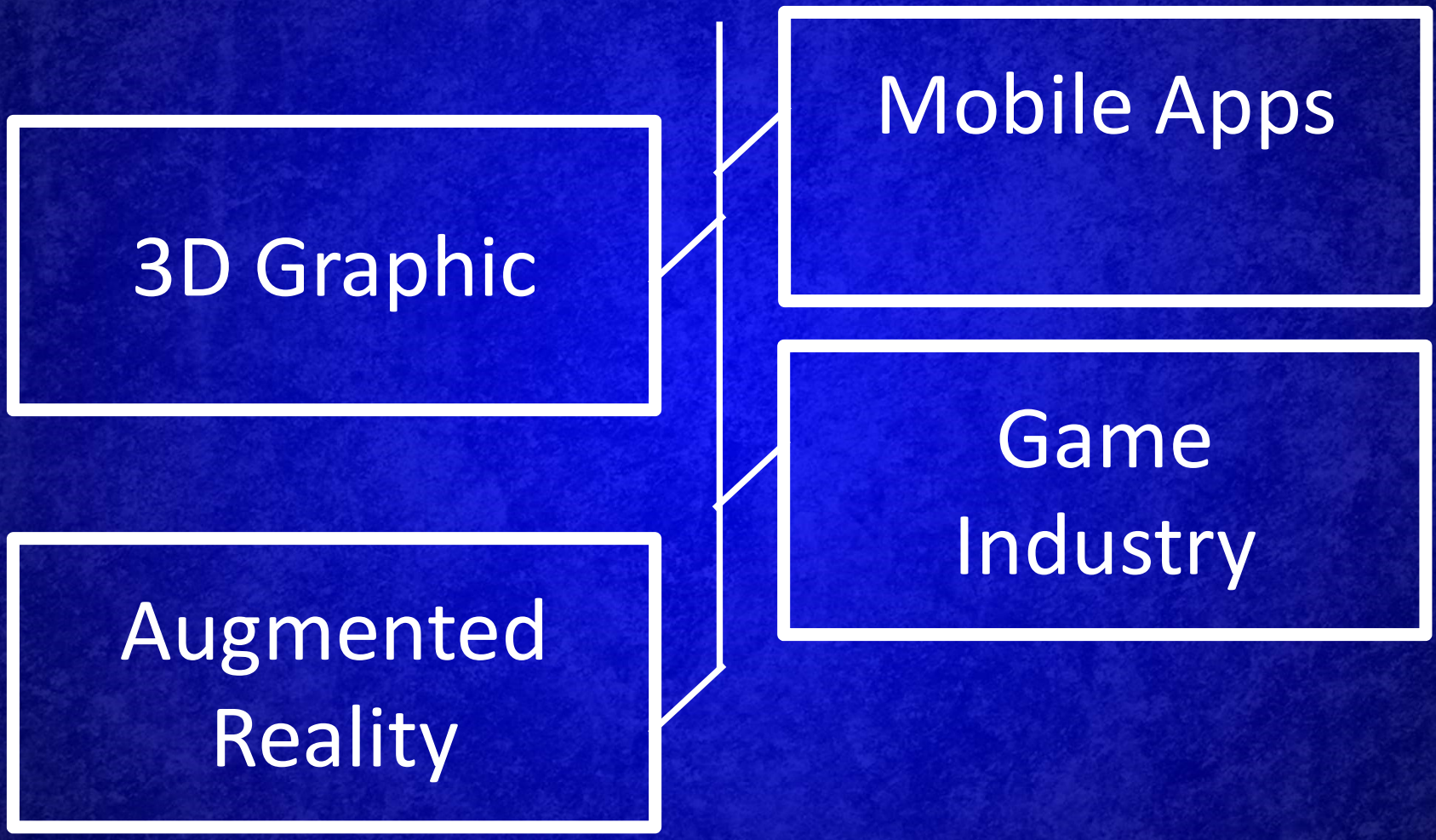
# SANTASCO

*"...SCIENTIA EST POTENTIA..."*

Software Developing
3D Graphics
Augmented Reality
Games
Digital Entertainment

# SANTASCO



# Unlock Unity Water Asset potential as we did in our Game

# Index

- Water shaders: basic parts
- Water shaders: advanced parts
- Common issues
- Performance impact
- Case study: Unity's Prefab
- Conclusion

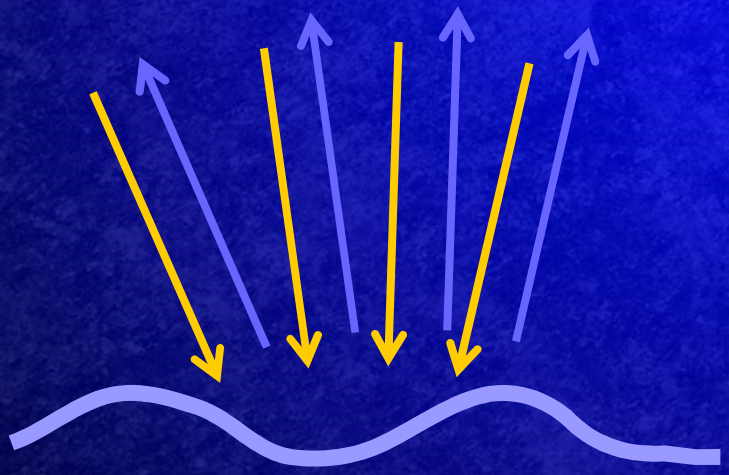
# Water shaders : basics

**surface_color = ambient + diffuse + reflection + refraction**

## - Ambient

Recive light from all directions and return it in the same all directions according to own coefficient

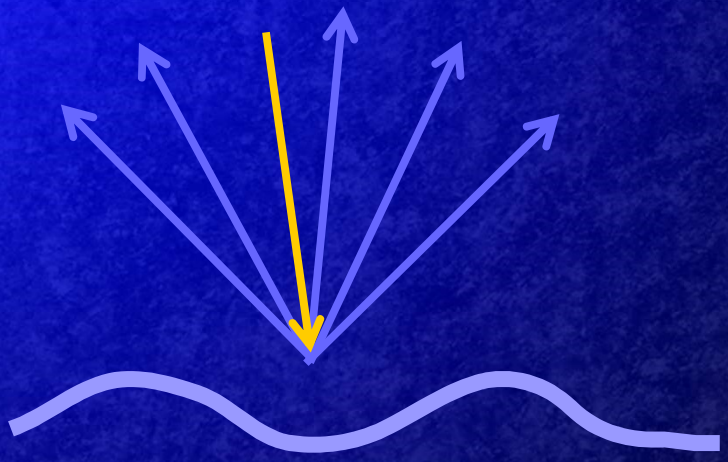
ambient = kA*global_ambient



## - Diffuse

Recives light from a single direction and reflects it equally in all directions

diffuse = kD x light_c * dot(N,L)



# Water shaders : basics

**surface_color = ambient + diffuse + reflection + refraction**

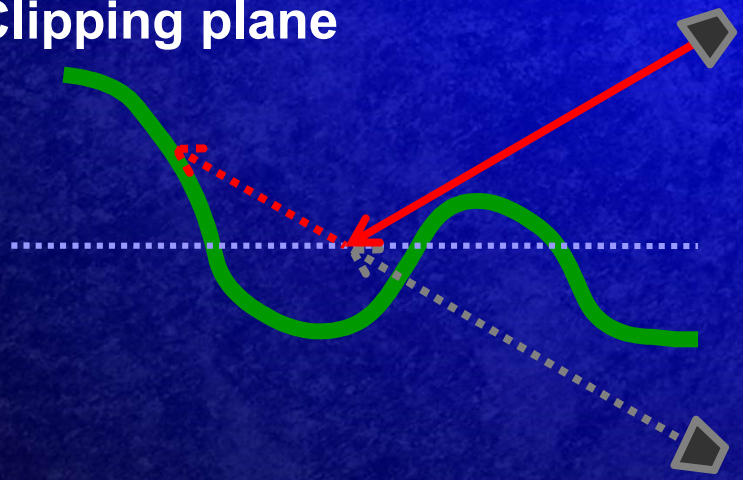
## - Reflection

Property to reflect the scene above the surface

The image of the reflected scene is needed to calculate the reflected color for each pixel of the water.

Extra render target texture

Clipping plane



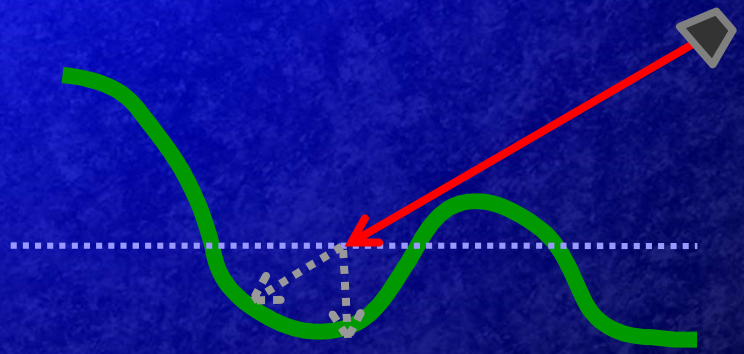
## - Refraction

Ammount of light acrossing the border of different density materials

Snell Law

Extra render target texture

Clipping plane

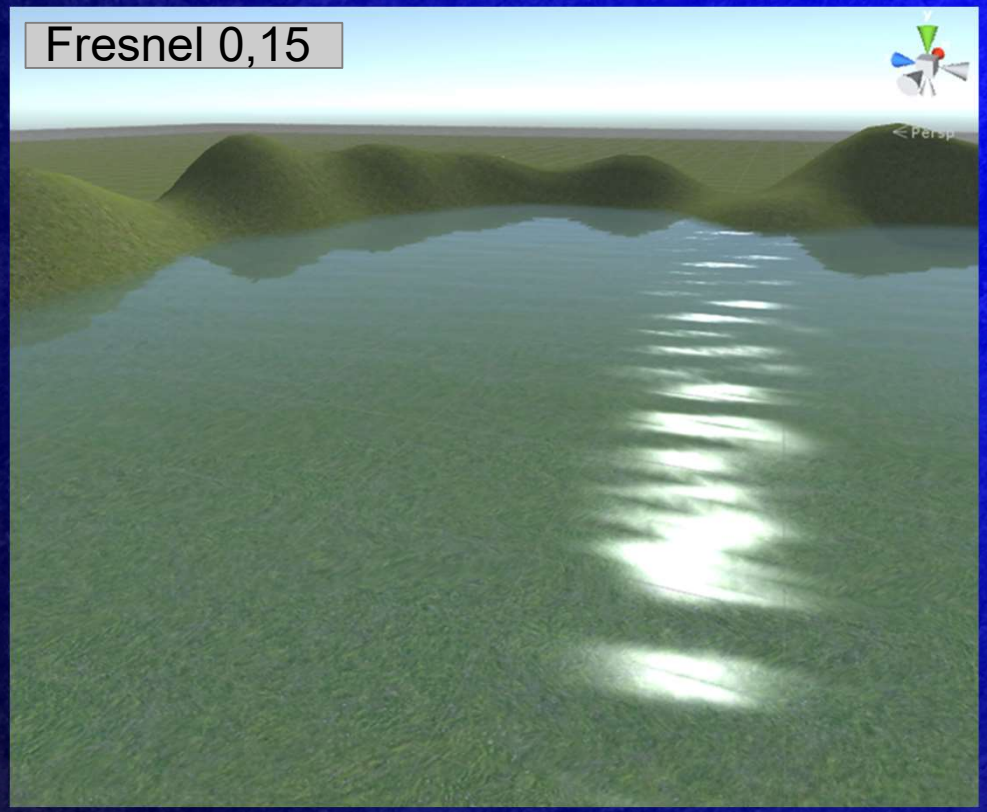


# Water shaders : basics

**surface_color = ambient + diffuse + reflection + refraction**

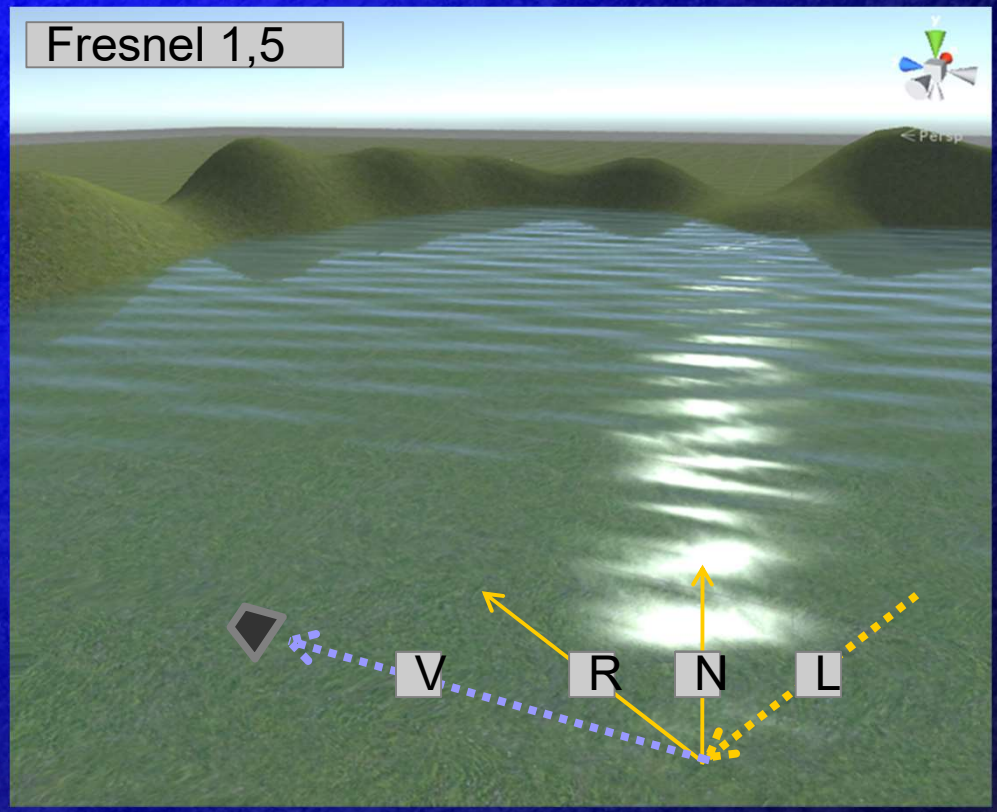
## - Fresnel

**Property to define reflection / refraction ratio**




## - Speculars

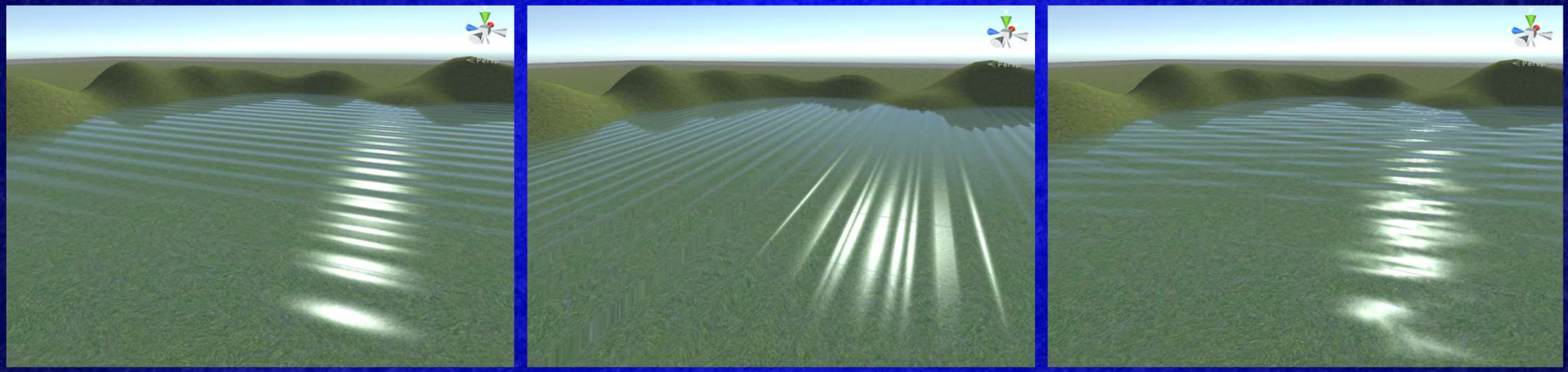
**High-lighted (sun) surface areas obtained by Phong illumination Model**



# Water shaders : basics

## - Waves

Achieved by scrolling in two different directions a bump map

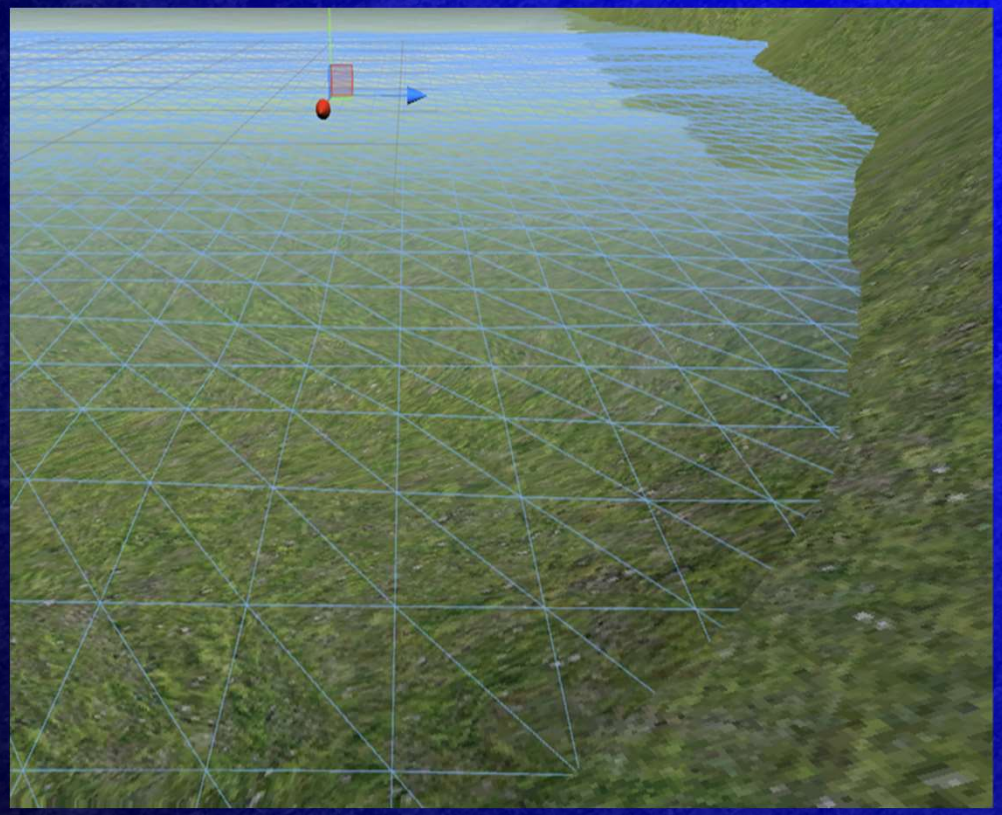


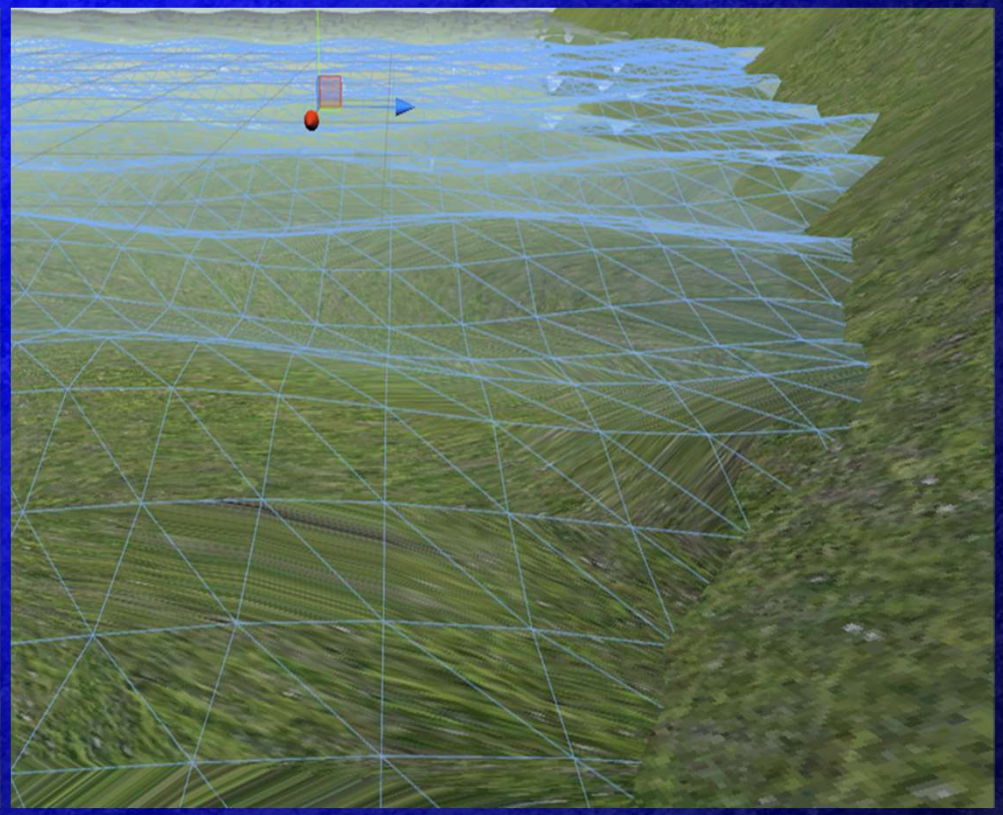
DIRECTION A + DIRECTION B = WAVES

# Water shaders : advanced parts
# - Solid waves by vertex displacement

amplitude
steepness
directionAB

frequency
speed
directionCD





# Water shaders : advanced parts

## - Solid waves by vertex displacement

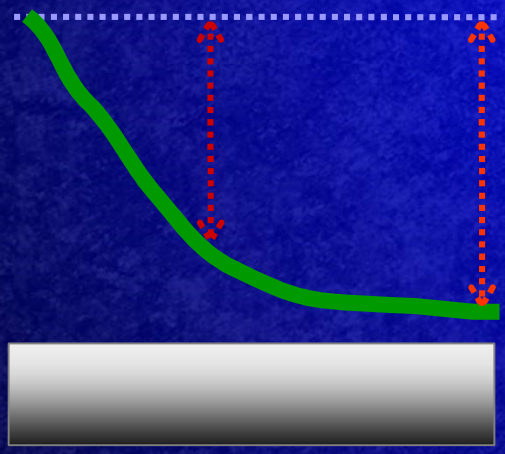
```
half3 GerstnerOffset4 (half2 xzVtx, half4 steepness, half4 amp, half4 freq, half4
    speed, half4 dirAB, half4 dirCD)
    {
    half4 AB = steepness.xxyy * amp.xxyy * dirAB.xyzw;     //DIR*STEP*AMP
    half4 CD = steepness.zzww * amp.zzww * dirCD.xyzw;

    half4 dotABCD = freq.xyzw * half4(dot(dirAB.xy, xzVtx), dot(dirAB.zw,     xzVtx),
    dot(dirCD.xy, xzVtx), dot(dirCD.zw, xzVtx)); //DOT PRODUCT DIR*FREQ
    half4 TIME = _Time.yyyy * speed;

    half4 COS = cos (dotABCD + TIME); //TIME MAKES IT MOVE
    half4 SIN = sin (dotABCD + TIME);

    offsets.x = dot(COS, half4(AB.xz, CD.xz));
    offsets.z = dot(COS, half4(AB.yw, CD.yw));
    offsets.y = dot(SIN, amp);  //FINAL COORDINATES

    return offsets;
}
```

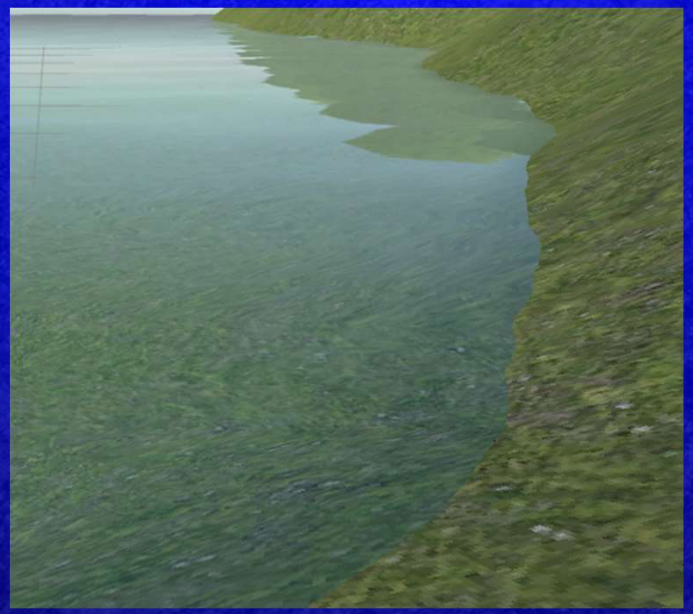
# Water shaders : advanced parts

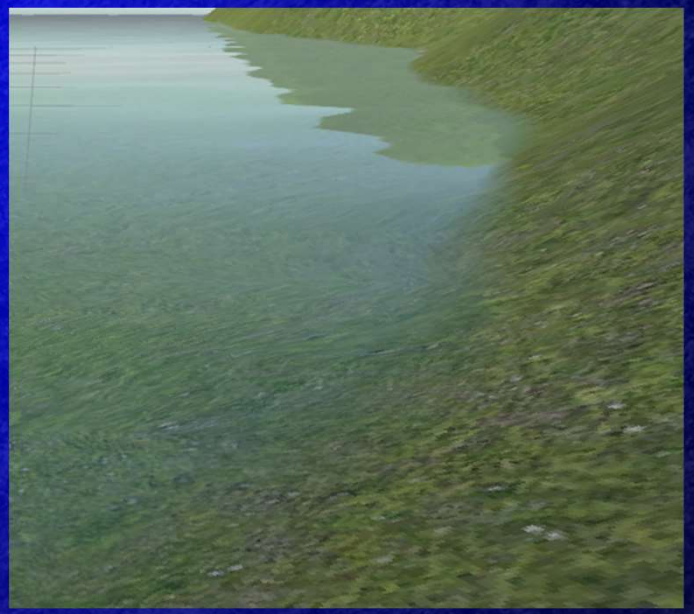
## - Edge Blending

Its purpos is to avoid hard edges where water plane intersect with anysurface

Achieved by calculate alpha of the pixel in linear function with the distant of the surface underwater



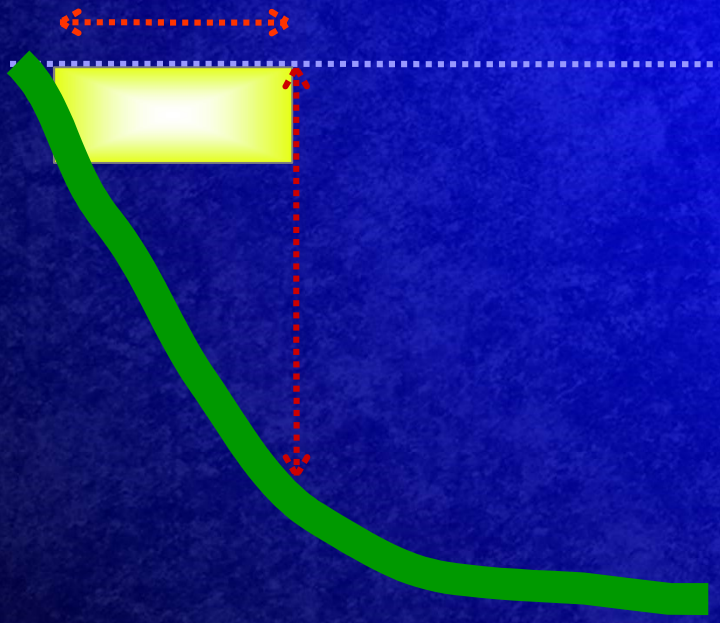


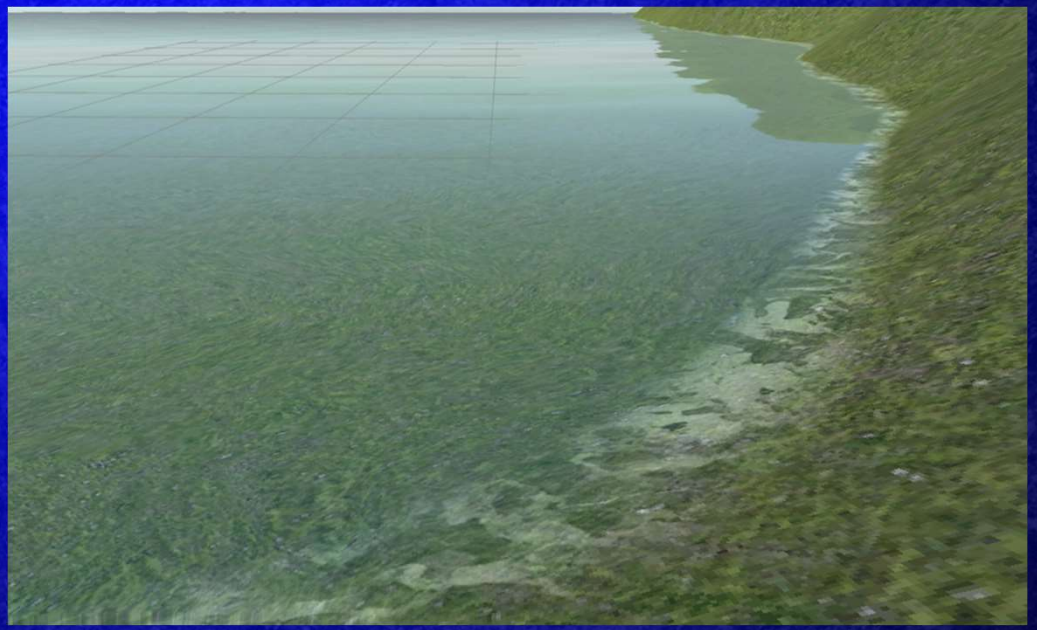


# Water shaders : advanced parts

## - Foam Blending

**Extra foam texture is added for a fixed distance starting from the edge intersection alpha blended at both sides**





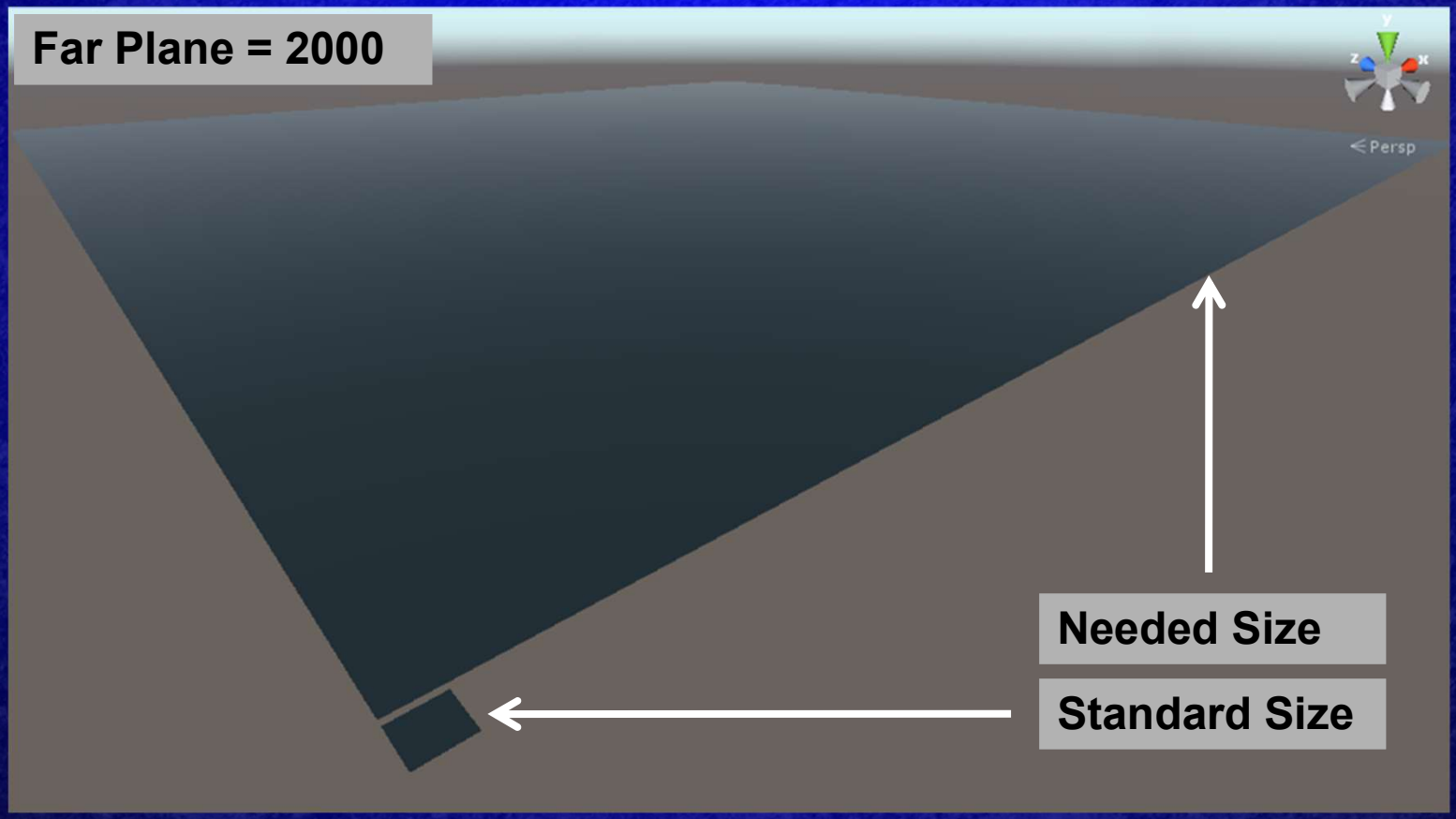
# Common Issues: plane size

**SCALE : BAD IDEA**

Results in deforming waves shape and size

**MORE PLANES : MORE VERTEX**

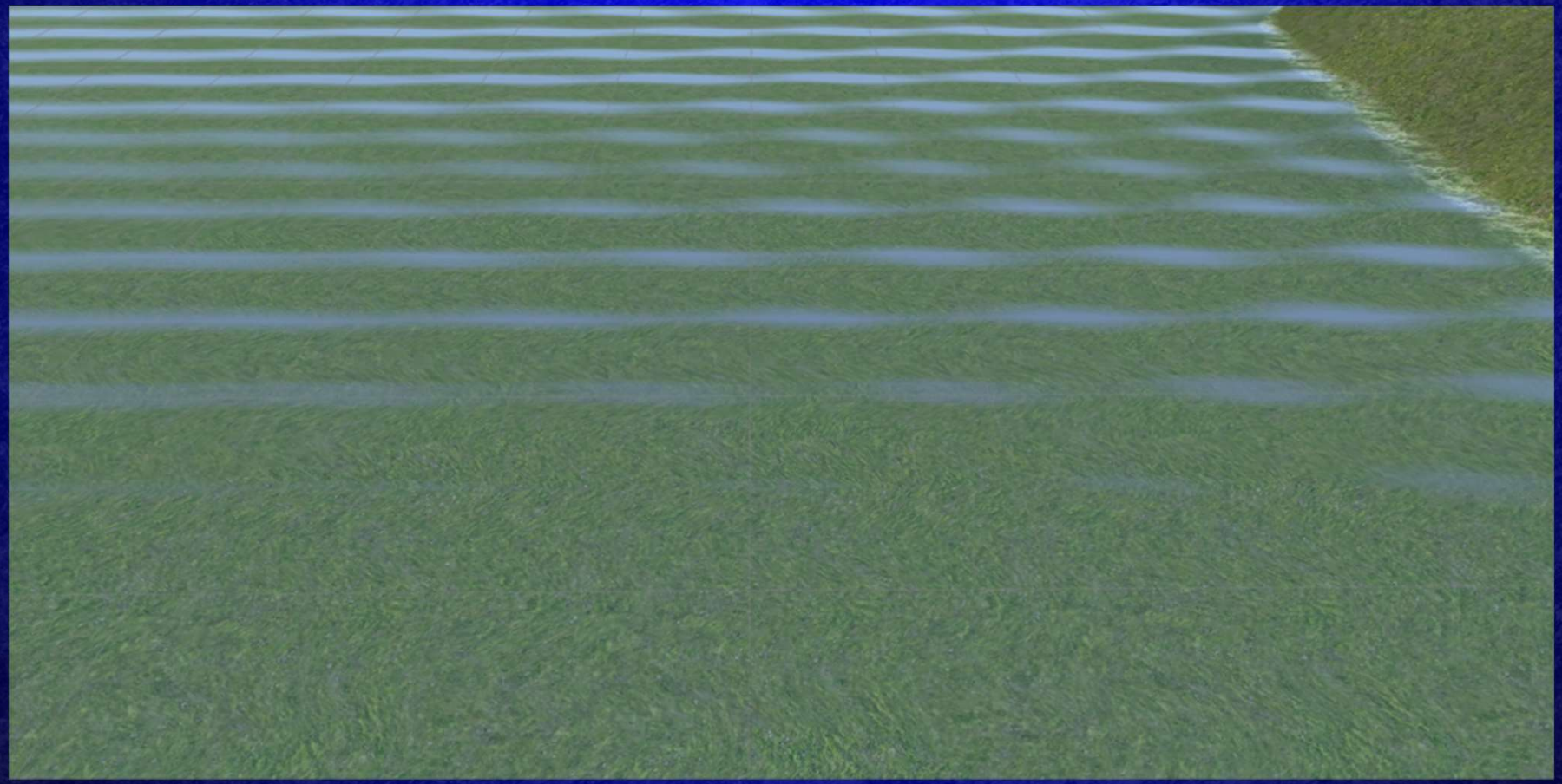
Drammatic increase of vertex count



# Common Issues

## - Wave pattern and repetition

**Unrealistic wave repetition**



# Common Issues

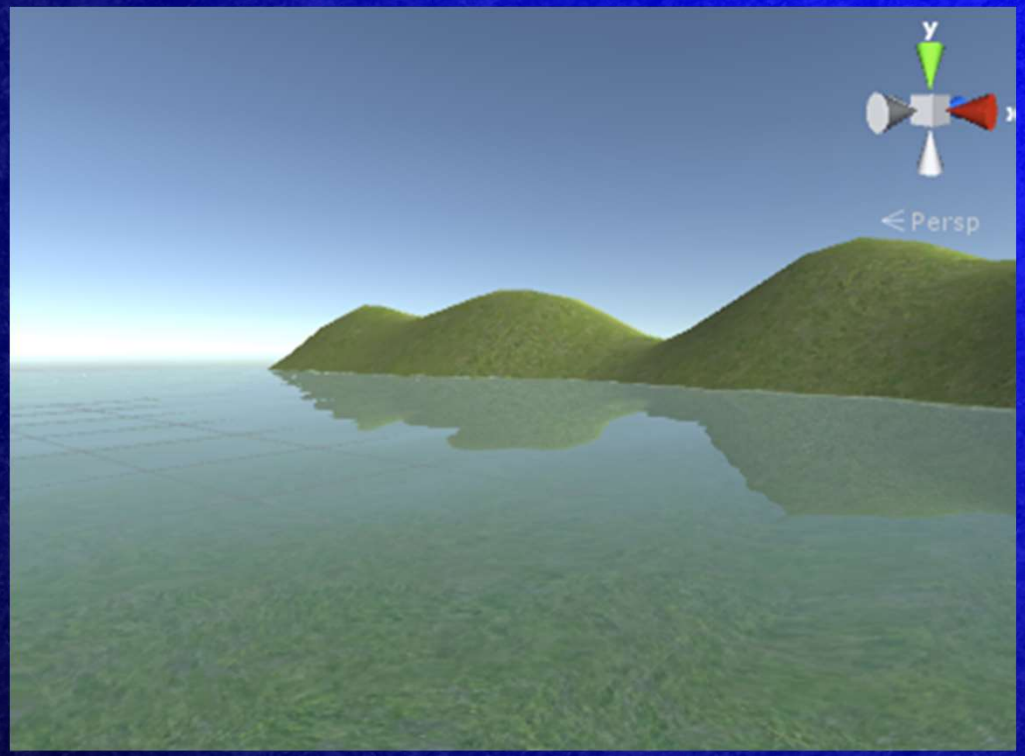
## - Culling or Overlay

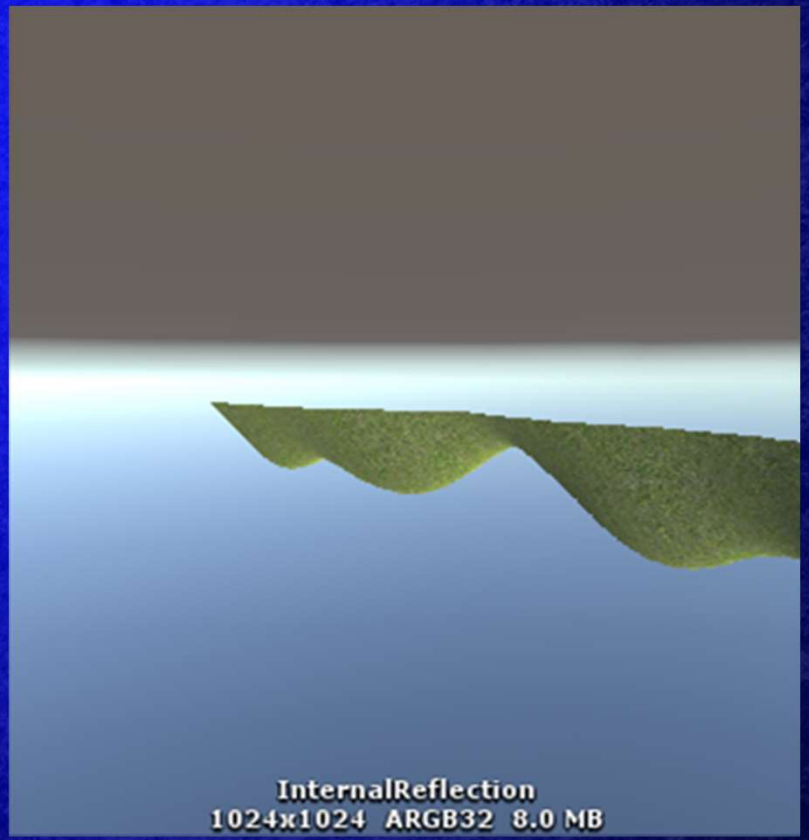
**Water is hidden or hides other object in the scene**

# Performance Impact

## - Reflection texture

Let's remember we are rendering an EXTRA SCENE

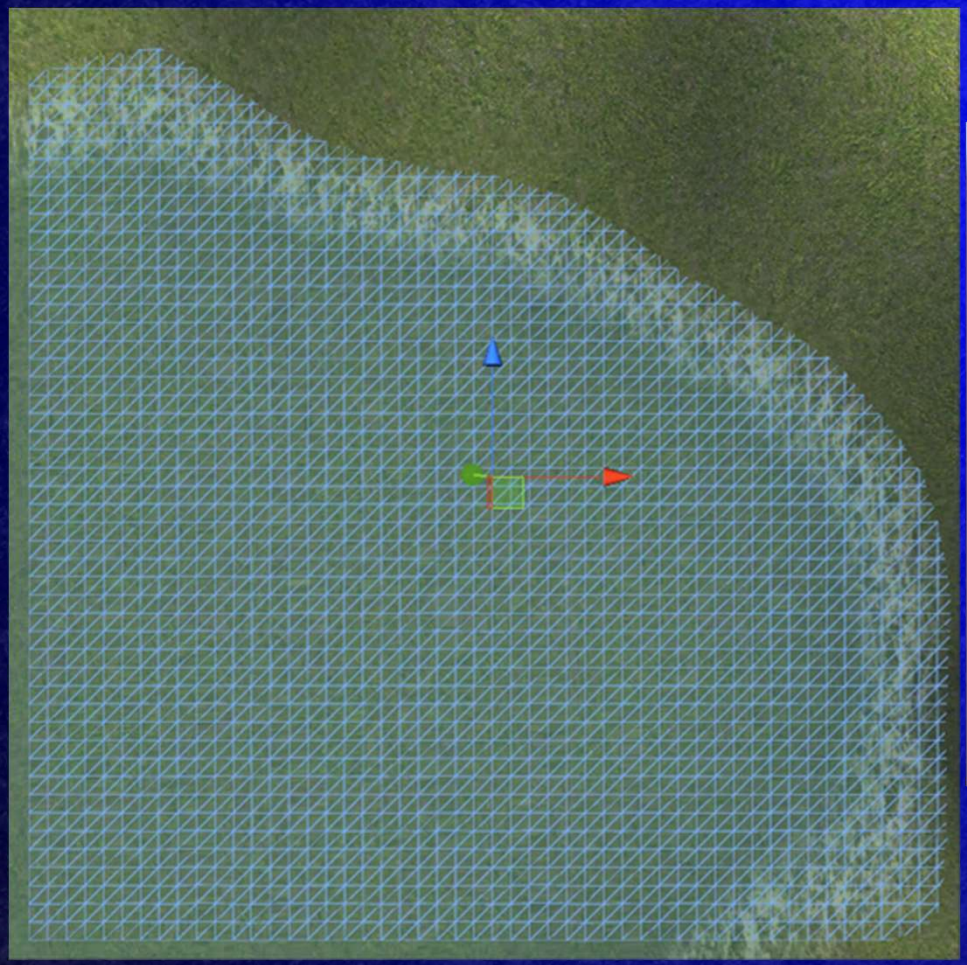


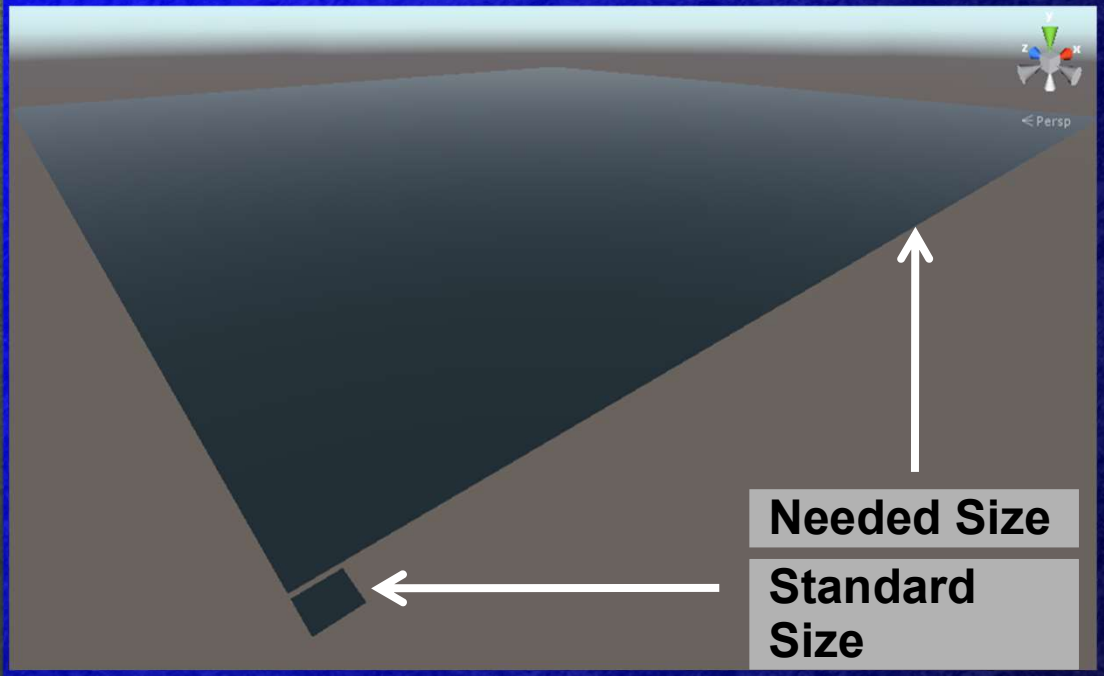



# Performance Impact

## - Vertex Density

This plane has 2652 vertex and it cover ¼ of the standard little plane





# Case Study: Solve pattern repetition

```
half3 GerstnerOffset4 (half2 xzVtx, half4 steepness, half4 amp, half4 freq, half4 speed, half4 dirAB,
    half4 dirCD)
    {
    half3 offsets;

    half4 AB = steepness.xxyy * amp.xxyy * dirAB.xyzw;
    half4 CD = steepness.zzww * amp.zzww * dirCD.xyzw;

    half4 dotABCD = freq.xyzw * half4(dot(dirAB.xy, xzVtx), dot(dirAB.zw, xzVtx), dot(dirCD.xy,
    xzVtx), dot(dirCD.zw, xzVtx));
    half4 TIME = _Time.yyyy * speed;

    half4 COS = cos (dotABCD + TIME);
    half4 SIN = sin (dotABCD + TIME);

    offsets.x = dot(COS, half4(AB.xz, CD.xz));
    offsets.z = dot(COS, half4(AB.yw, CD.yw));

    offsets.y = dot(SIN, amp) + rand(float3(xzVtx.x, xzVtx.y, 0))*0.2 - 0.1;

    return offsets;
    }
```

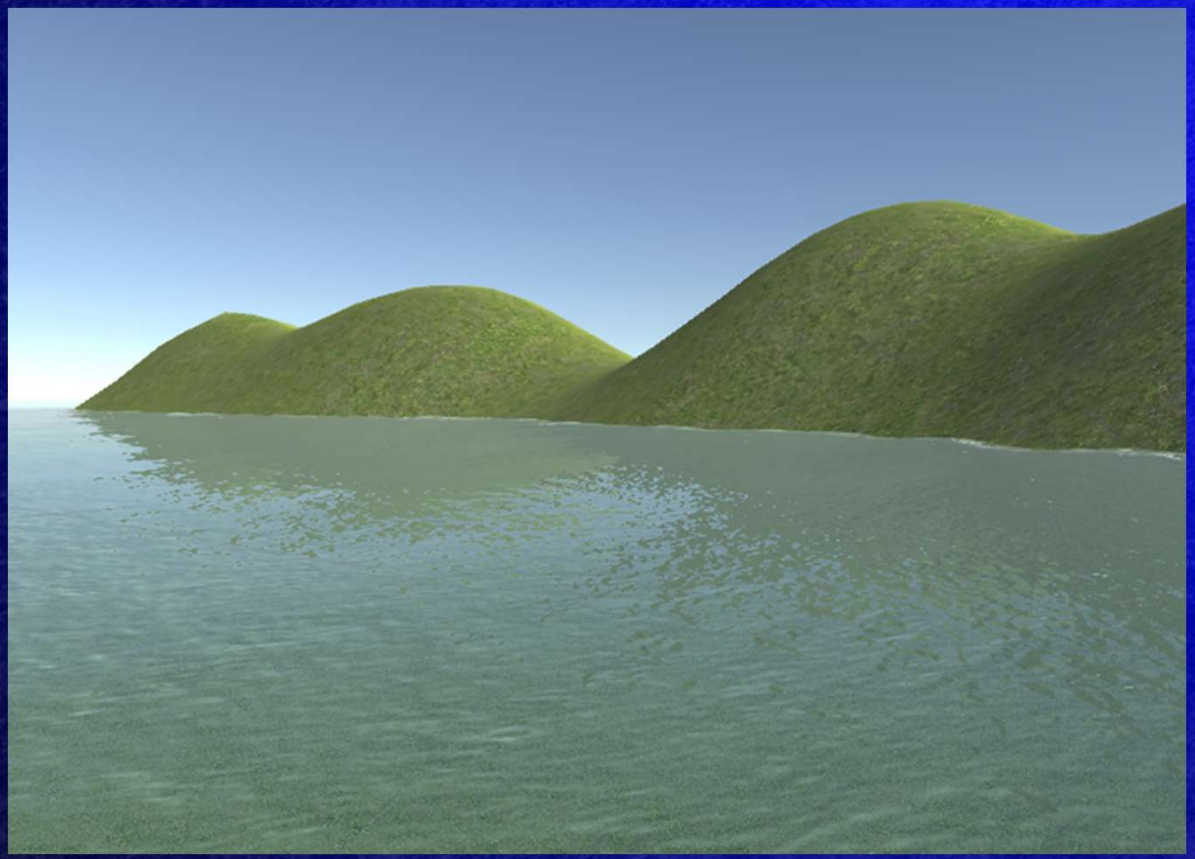
# Case Study: Solve Culling or overlay

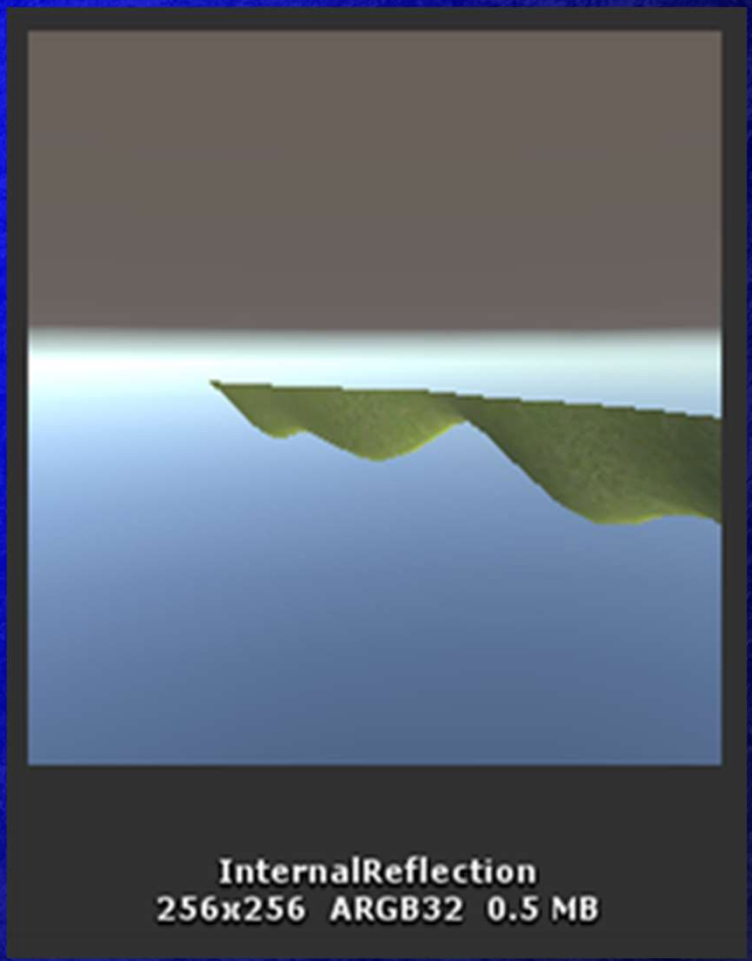
## Cause:Position in the render order

```
Subshader
{
    Tags {"RenderType"="Transparent" "Queue"="Transparent-200"}

    Lod 500
    ColorMask RGB

    GrabPass { "_RefractionTex" }

    Pass { ...
```

# Case Study: Limit Reflection Texture performance impact

-Resize the texture as small as possible
-Render only essential layers or change according distance
-Add an extra scrolling bump map to add extra noise

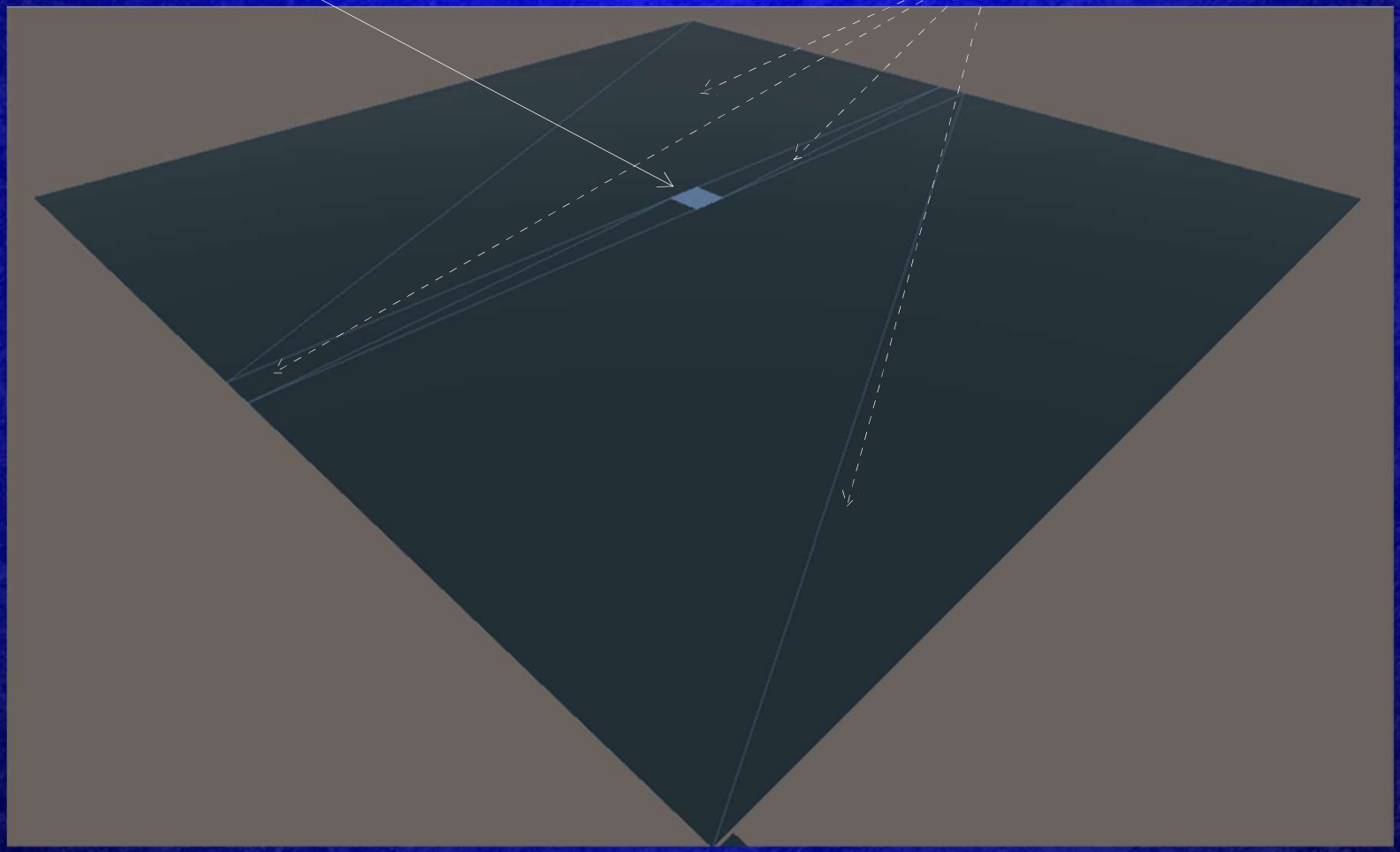




# Case Study: Achieve huge size & keep low vertex count

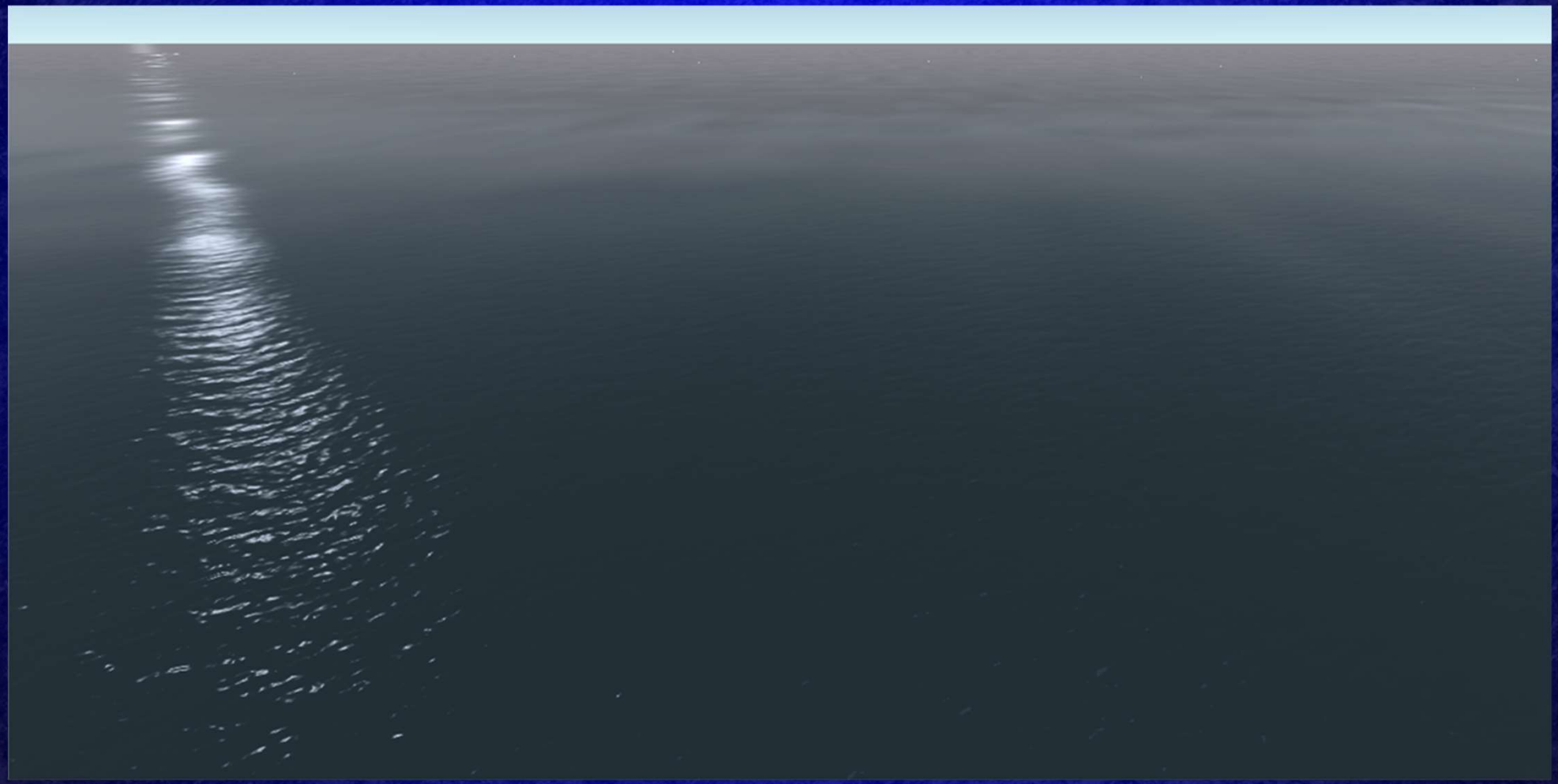
Detailed waved plane matrix 32 x 32

Simple 4 vertex planes



# Case Study: Achieve huge size & keep low vertex count

The detailed plane matrix and must have the same shader with exactily the same properties except WAVE AMPLITUDE that MUST BE 0 FOR QUADS

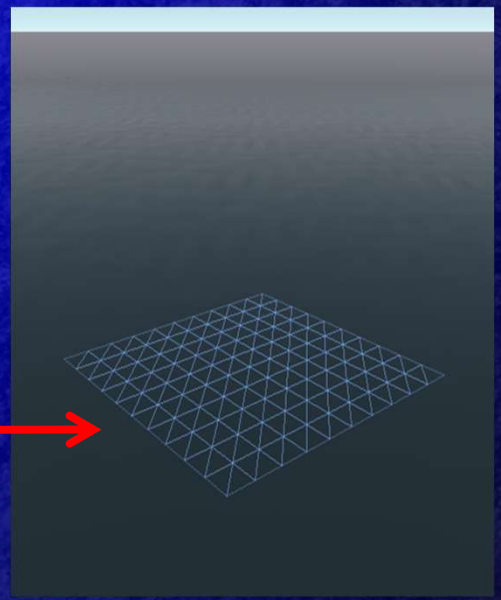


# Case Study: Achieve huge size & keep low vertex count

**What about camera movements?**

# Case Study: Achieve huge size & keep low vertex count

```
dist = Vector3.Distance(ekr.transform.position,this.transform.position);
    xDist = ekr.transform.position.x - this.transform.position.x;
    yDist = ekr.transform.position.z - this.transform.position.z;

    if( Mathf.Abs(dist) > maxDist )
    if( Mathf.Abs(xDist) > Mathf.Abs(yDist) )
    {
     this.transform.position = this.transform.position +
    Vector3.right*minStep*Mathf.Sign(xDist);
    }
    else
    {
     this.transform.position = this.transform.position +
    Vector3.forward*minStep*Mathf.Sign(yDist);
    }
```



# Conclusions

- Perfect for high far plane value and collisions
- Simple and Customizable
- Performance proof
- Applicable on any engine
- Unity case: 100%versions compatible no external components

# Thank you

www.santasco.com