The Ocean System in Pirates of the Caribbean


Disney
加勒比海盗
荣耀之海
西山居

III级 护航战列舰
100/100
146.0公里
18
M墨菲
登录豪礼
活动
福利
公海
莱凯
(3272, 1185)
旅行模式
6.5kt
系统 获得了 [木板] x 35
系统 获得了 [木板] x 2
系统 获得了 50 银币
邮件
好友
背包
协会
15:15
700/1000

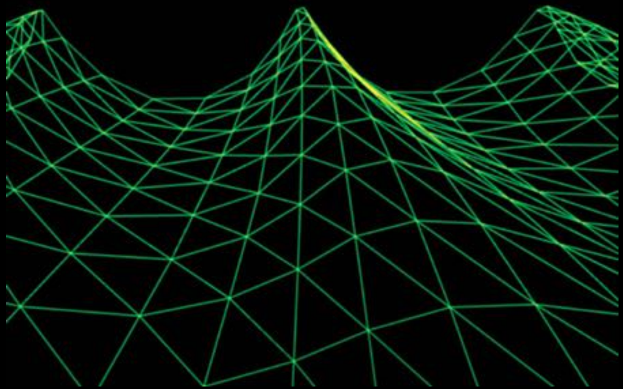
Ocean in Pirates of the Caribbean

- Motivation
- Implementation
  - Wave animation
  - Ocean surface material
  - Interactions between water surface and ships
- Fit for mobile devices

Ocean in Pirates of the Caribbean

The basic implementation of an ocean surface

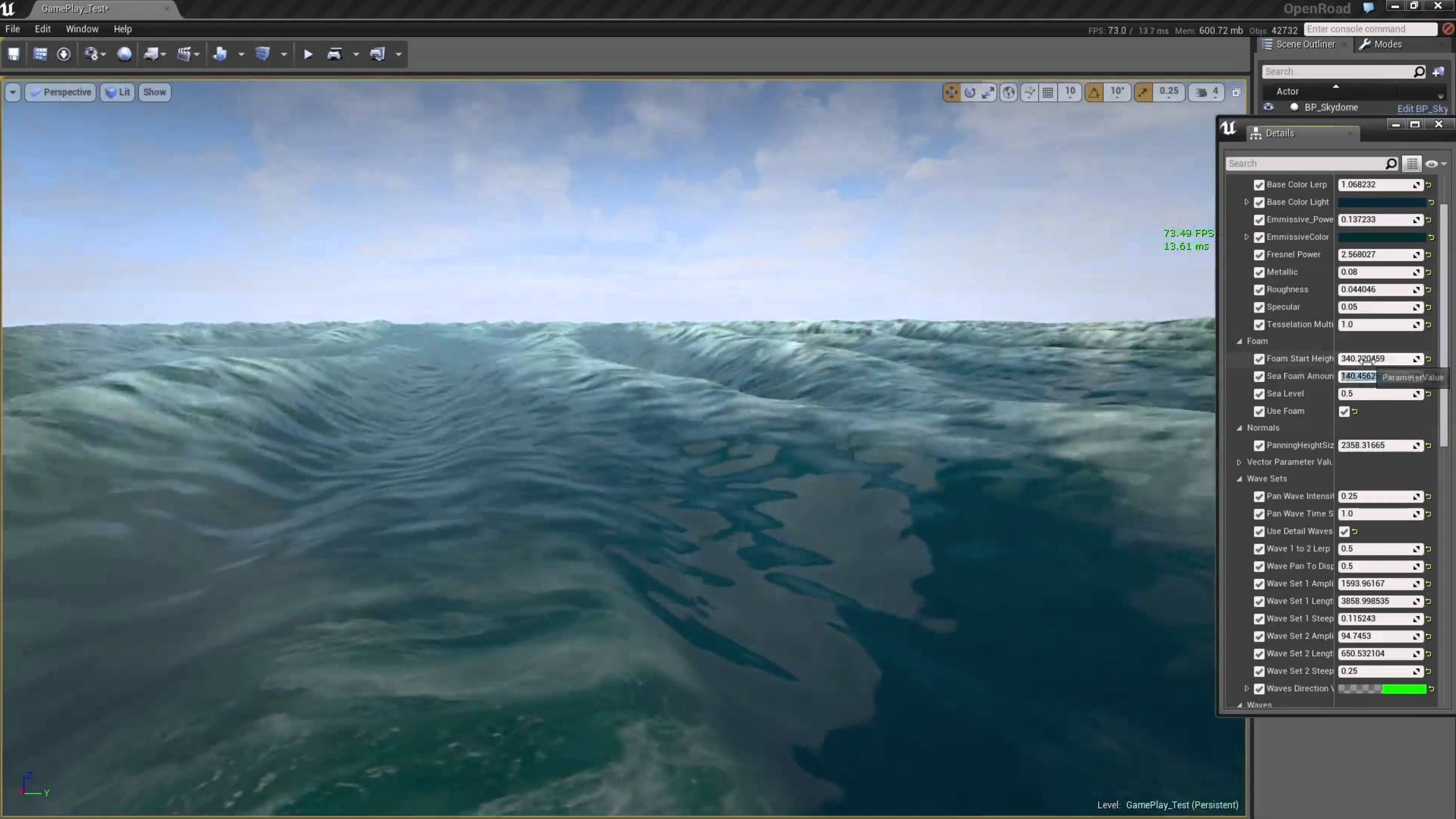
- With a grid mesh
- Drive the vertices move along some curves. (Such as Sin curve)
- Make some improvements on the sin curve, like Gerstner Wave. (GPU Gems 1 Chapter 01)



Implementation method before

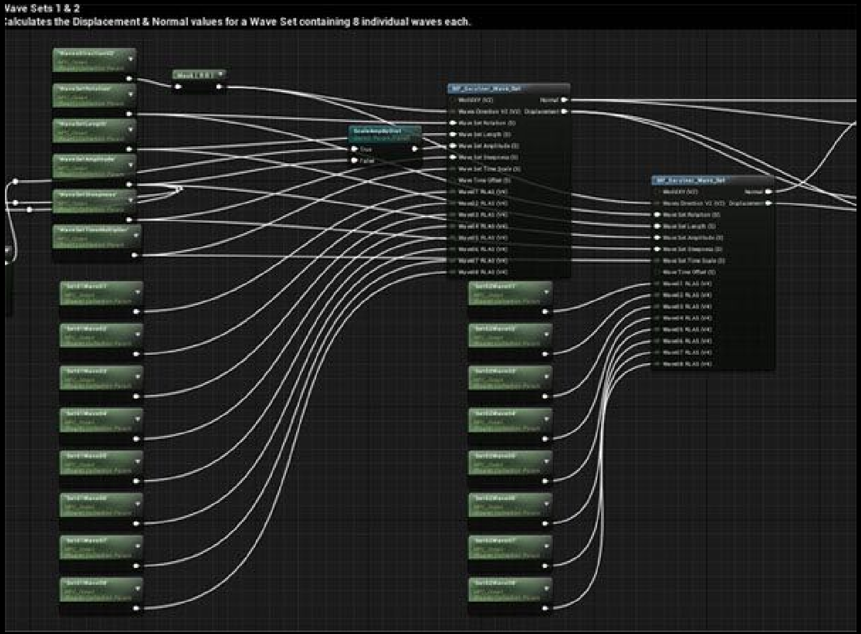
We use Gerstner Wave, and get an effect like this

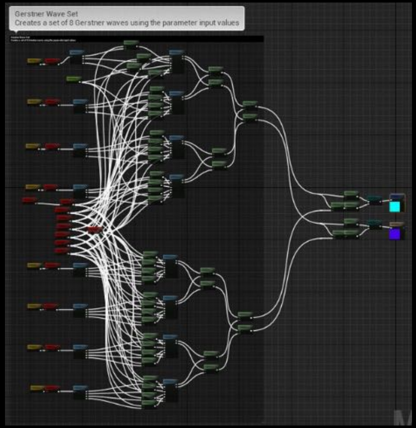

I级 双桅快船
95/100
98.0公里
[支线] 港口骚乱
与码头管理员对话
任务
队伍
理查德·瓦伦
基诺·刘易斯
家港
公海 2957,1295
30.4kt

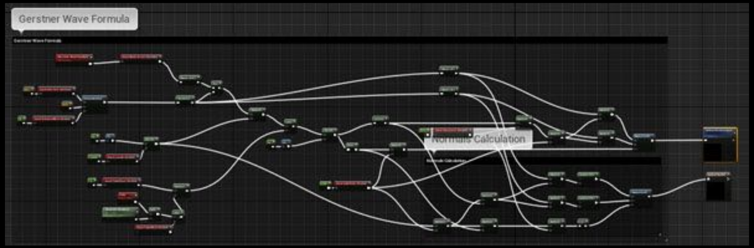

GamePlay_Test*
OpenRoad
File
Edit
Window
Help
FPS: 73.0 / 13.7 ms Mem: 600.72 mb Objs: 42732
Enter console command
Scene Outliner
Modes
Search...
Actor
BP_Skydome
Edit BP_Sky
Perspective
Lit
Show
10
10°
0.25
4
73.49 FPS
13.61 ms
Details
Search
Base Color Lerp 1.068232
Base Color Light
Emmissive_Powe 0.137233
EmmissiveColor
Fresnel Power 2.568027
Metallic 0.08
Roughness 0.044046
Specular 0.05
Tesselation Multi 1.0
Foam
Foam Start Heigh 340.220459
Sea Foam Amoun 140.4562
ParameterValue
Sea Level 0.5
Use Foam
Normals
PanningHeightSiz 2358.31665
Vector Parameter Valu
Wave Sets
Pan Wave Intensi 0.25
Pan Wave Time S 1.0
Use Detail Waves
Wave 1 to 2 Lerp 0.5
Wave Pan To Disp 0.5
Wave Set 1 Ampli 1593.96167
Wave Set 1 Lengt 3858.998535
Wave Set 1 Steep 0.115243
Wave Set 2 Ampli 94.7453
Wave Set 2 Lengt 650.532104
Wave Set 2 Steep 0.25
Waves Direction
Waves
Level: GamePlay_Test (Persistent)

Gerstner Wave

Shader Graph







Gerstner Wave

16 waves calculations overlap with each other.

4 directions of detail normals for surface details.

Advantages

- Calculation at runtime, do not need baked resources

Disadvantages

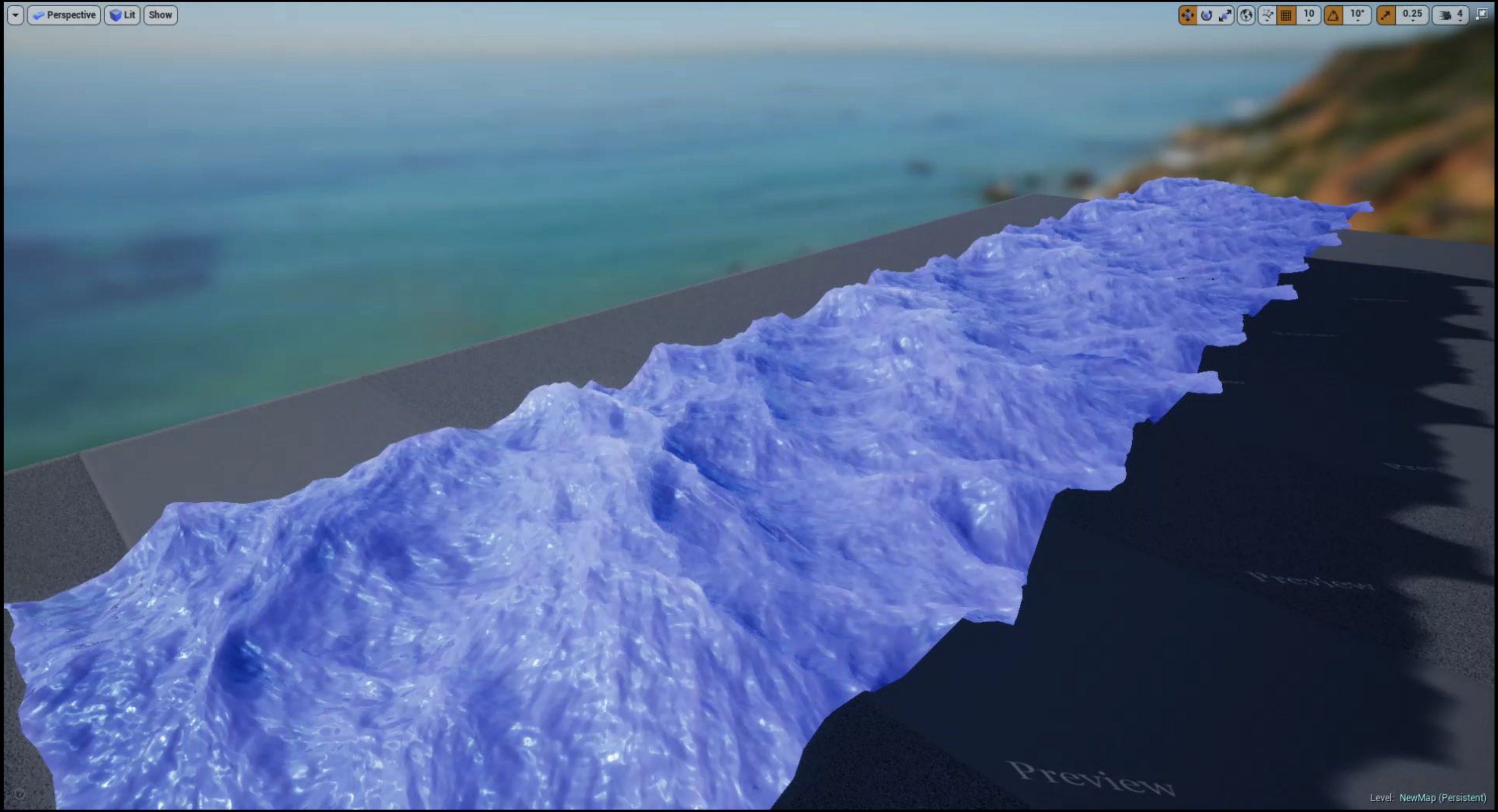
- Very complex parameters
- Very few ocean waves (16 Max), do not have a great visual effect
- Real-time shader calculation is heavy

Gerstner Wave

And we found this image.

FFT Ocean

- The classic ocean simulation paper by Jerry Tessendorf. “Simulating Ocean Water”
- Based on Fast Fourier Transform algorithm.
- 32x32 to 1024x1024 groups of ocean waves


Perspective
Lit
Show
10
10°
0.25
4
Preview
Preview
Level: NewMap (Persistent)

FFT Ocean

Advantages:

- A lot of wave details, have a much better look of ocean surface.
- With only a few parameters to get the correct effect.

Disadvantages:

- Algorithm is complicated. CPU can not afford the real-time FFT calculation even on PC. Some developers move it to GPU to speed up for real-time calculation
- Mobile systems on different devices have a very bad compatibility for compute shader in ES3.1

FFT Ocean implementations for mobile

A FFT Ocean driven by a flip book

- Bake the FFT calculation result to flip books
- Without calculations, only need to load data when rendering in real time.

Advantages:

- High-quality ocean waves simulated with FFT algorithms
- Avoid a lot of real-time computing (CPU & GPU)

Disadvantages:

- Pre-baked data is required.

Implementation of Ocean System

Flipbook FFT Ocean implementation on mobiles

- Wave animation
- Ocean surface material
- Sailing and water surface interaction

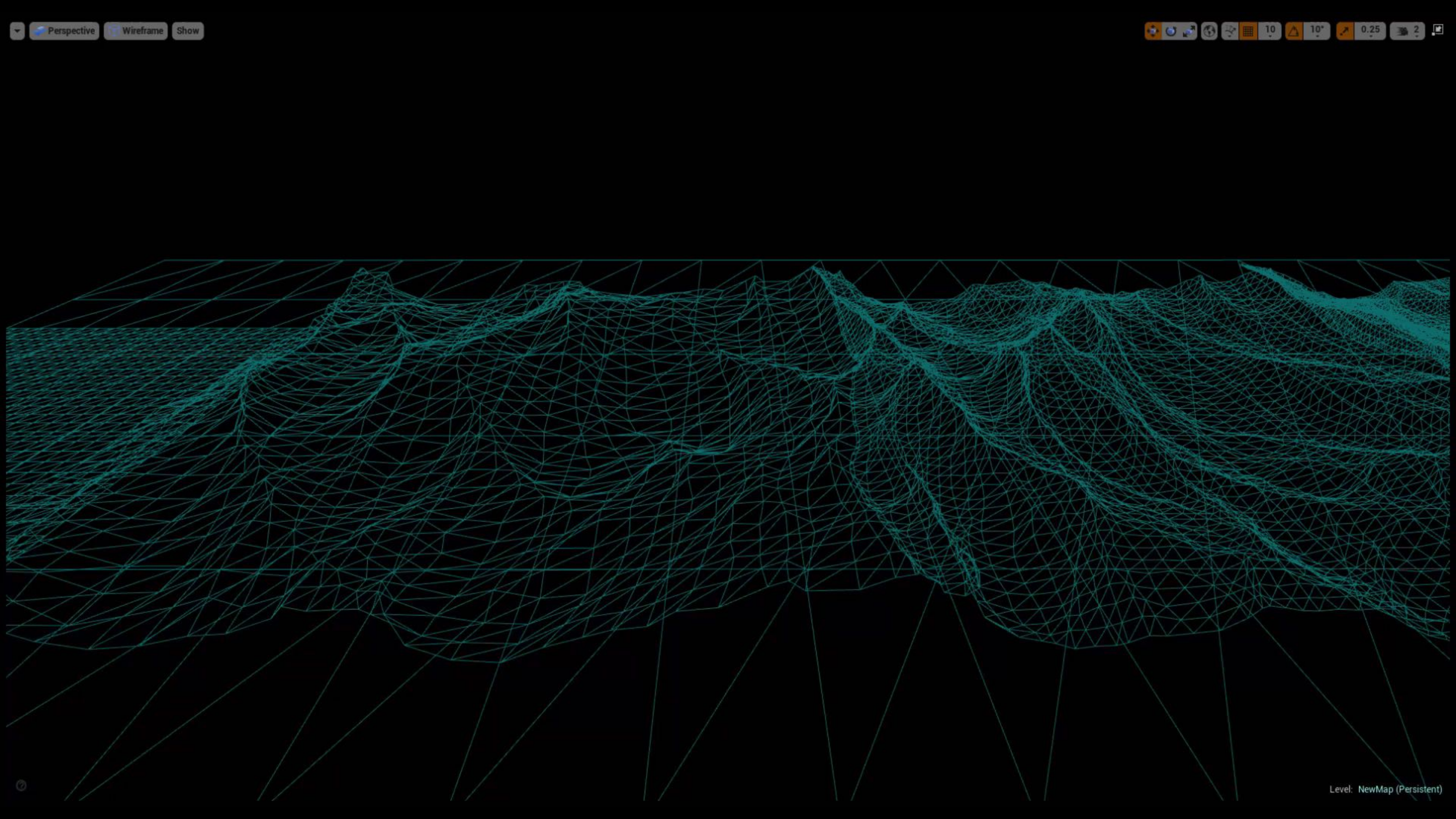
Implementation of Ocean System

Flipbook FFT Ocean implementation on mobiles

- **Wave animation**
- Ocean surface material
- Sailing and water surface interaction

Wave Animation

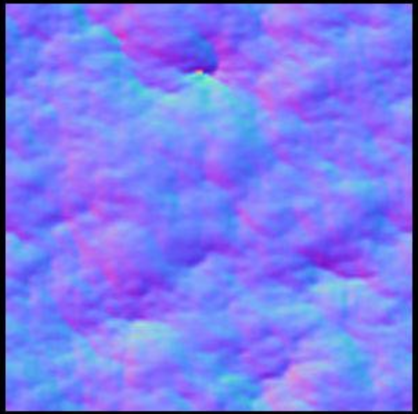
The deformation of the mesh can be seen as the displacement of each grid vertex along the X, Y, and Z axis.


Perspective
Wireframe
Show
10
10°
0.25
2
Level: NewMap (Persistent)

Wave Animation

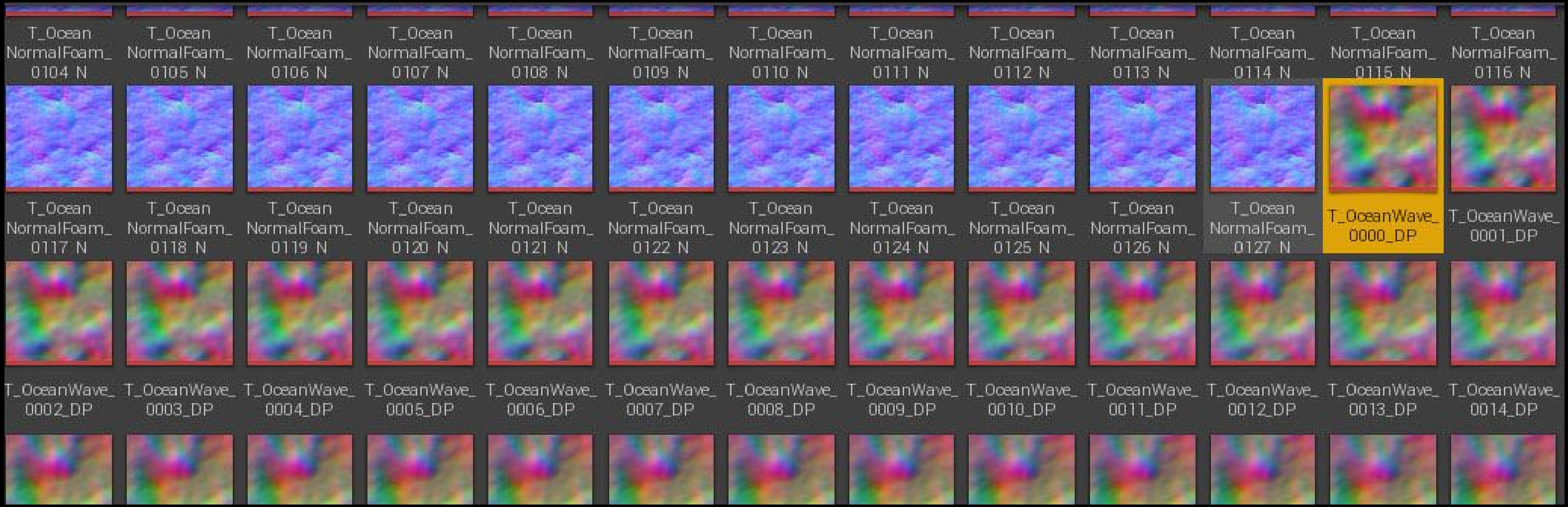
Ideas:

- Encode the displacement quantities of X,Y,Z to Channel R,G,B of a pixel, normalized.
- Encode the motion data of the vertices in a grid onto a texture.
- Each frame of animation data generates a texture, and then drives the vertex motion through a series of generated sequence frames.



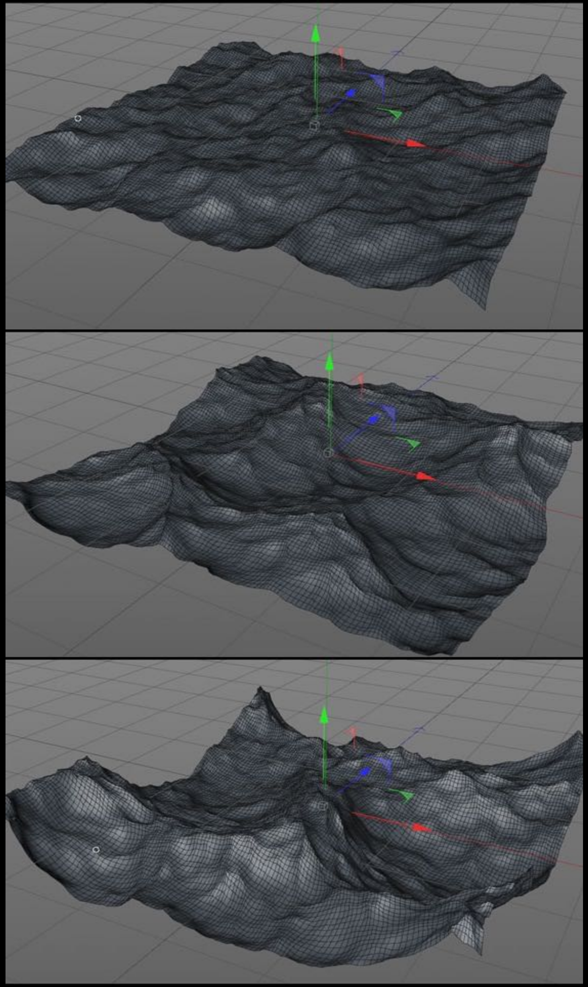
Wave Animation

That’s the flip books.



Wave Animation

- Load each frame at runtime to minimize load consumption
- Multiple sets of flipbooks can be switched at runtime



Wave Animation

Preprocessing data generation

- Use your custom algorithm
- Third-party DCC softwares
- Currently using **32x32** displacement maps, **128x128** normal maps with **128** sequence frames.

Wave Animation

- Created a custom vertex factory to animate with flipbooks
- Engine/Shaders/OceanVertexFactory.usf
- With es3.0 vertex texture fetch api
- For es2 system, replace the texture with a low precision buffer

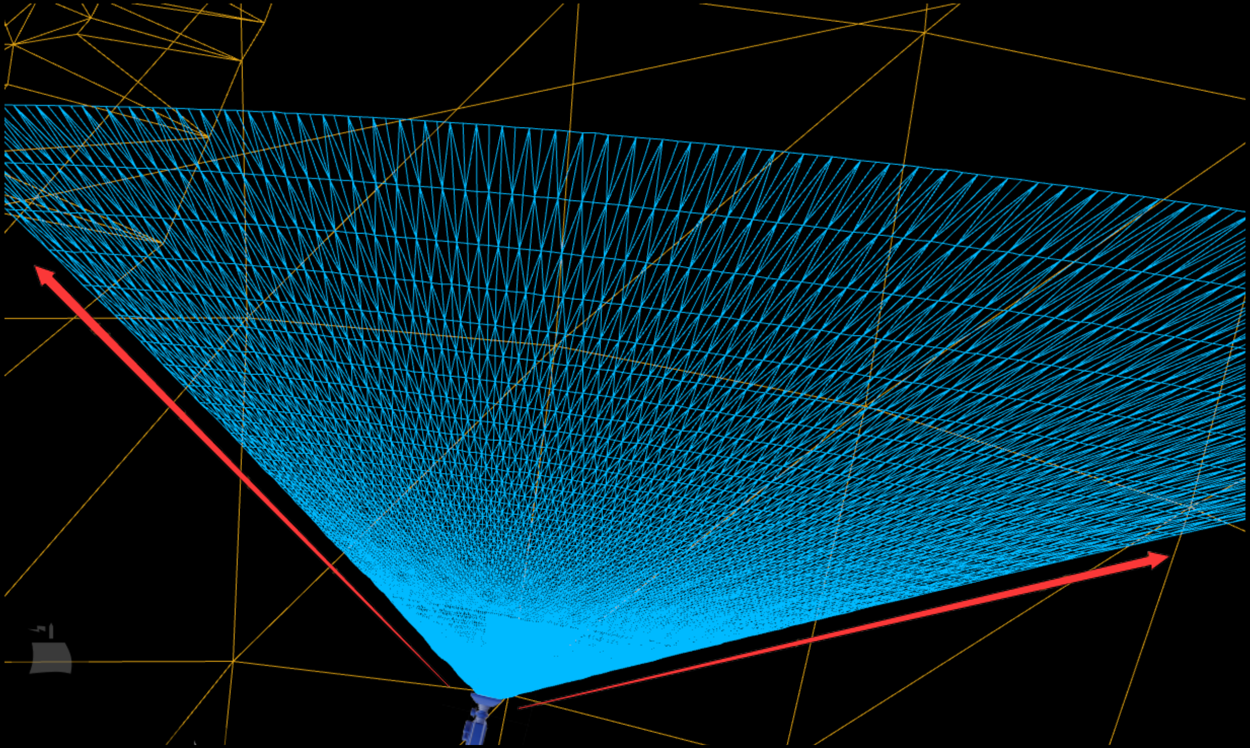
```
// Calculate each vertices' position offset
float4 WaveOffset = Texture2DSampleLevel( WaveTexture, WaveTextureSampler, WorldPosUVRot , 0 );
WaveOffset.xyz = WaveOffset.xyz * OceanWaveParams.WaveScale + OceanWaveParams.WaveStart;
```

Wave Animation: The Mesh

- Projection grid. High utilization and low redundancy calculation
- Quad-shaped topology. Suitable for ocean wave vertex animation
- Dense near, sparse far away. Maximize the use of triangles

- Reference to “Assassin’s Creed III: The tech behind (or beneath) the action”

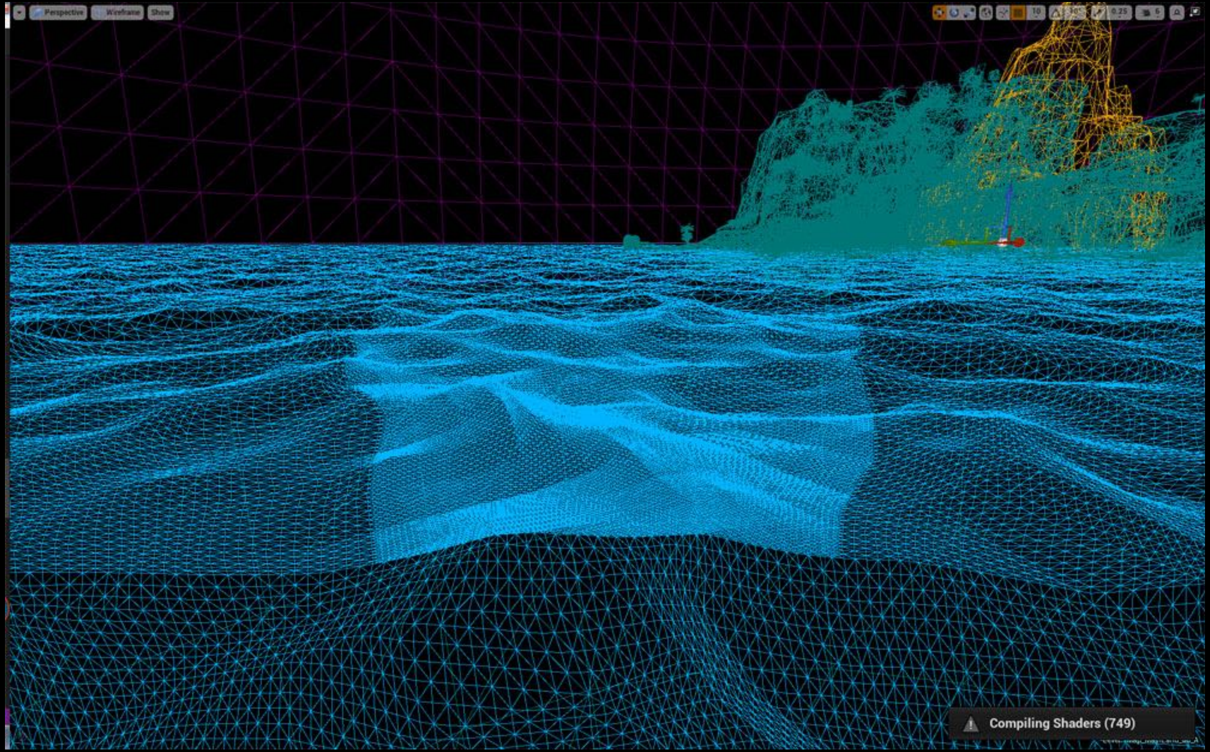
Wave Animation: The Mesh



Wave Animation: The Mesh

- 9-square distribution for easy LOD switching
- LOD switching based on map depth checking

Wave Animation: The Mesh

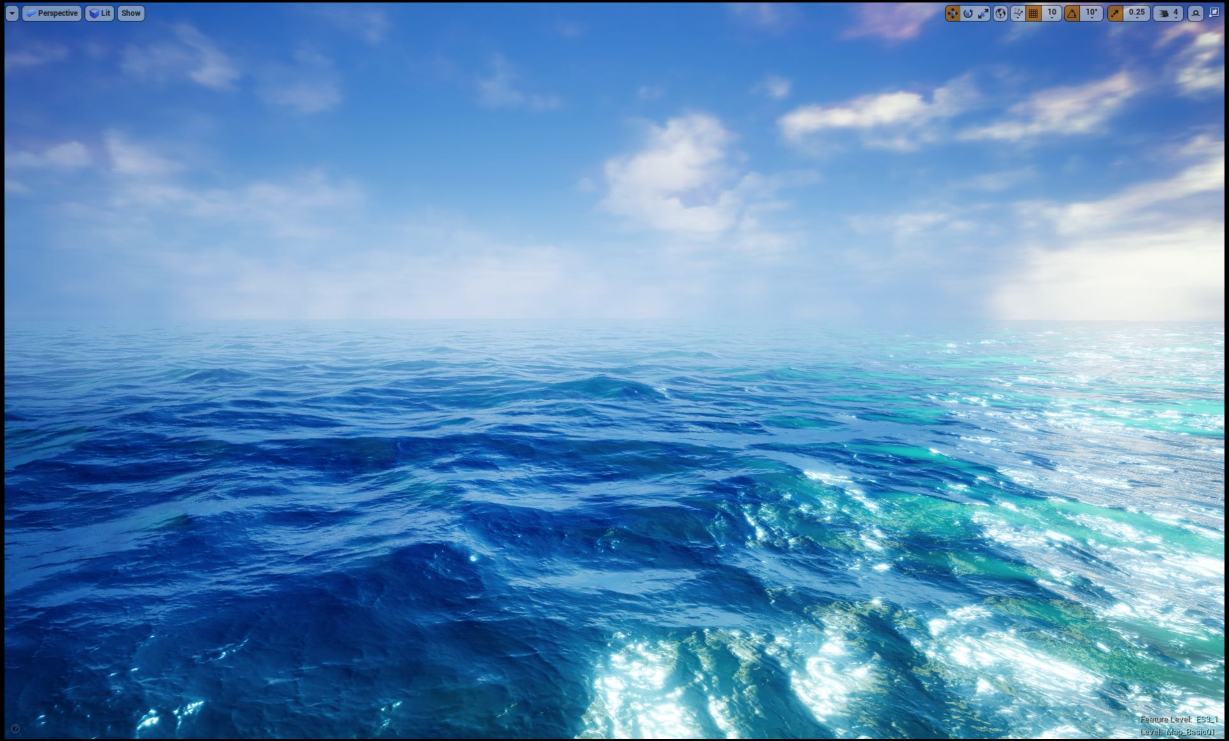


Implementation of Ocean System

Flipbook FFT Ocean implementation on mobiles

- Wave animation
- **Ocean surface material**
- Sailing and water surface interaction

Ocean Material : Base Color



Ocean Material : Base Color

Ocean Material : BRDF GGX vs Phong

Ocean Material : Planar Reflection

Ocean Material : Planar Reflection

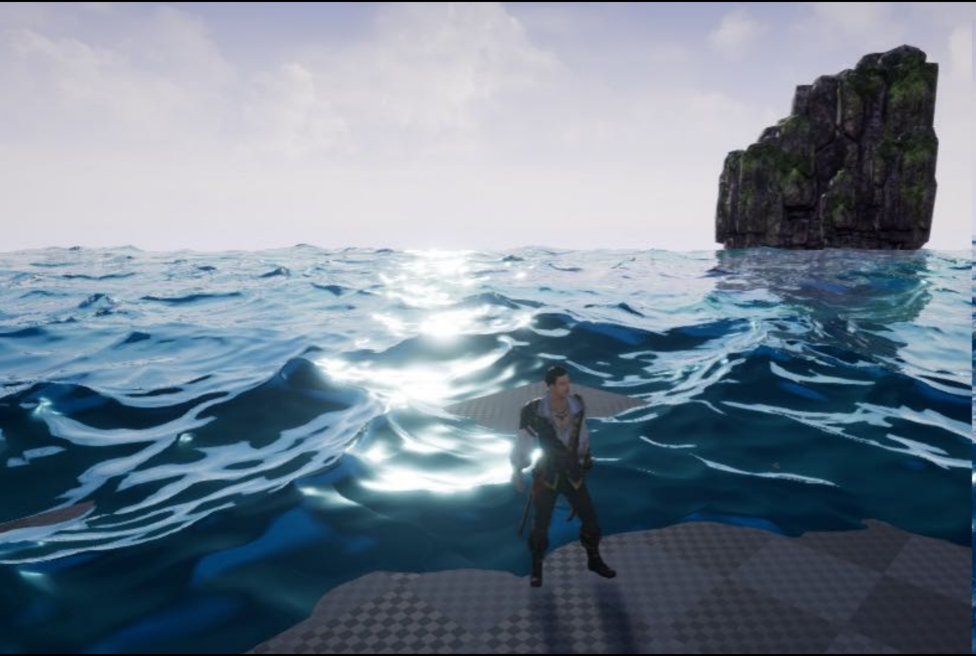
Ocean Material : Planar Reflection

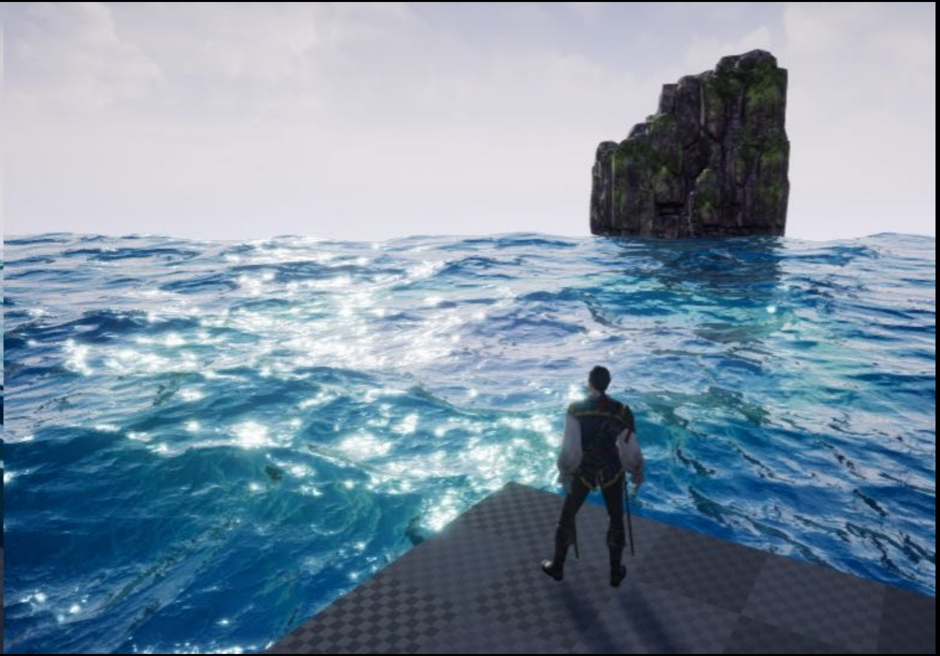
Ocean Material : Planar Reflection

Ocean Material : Planar Reflection

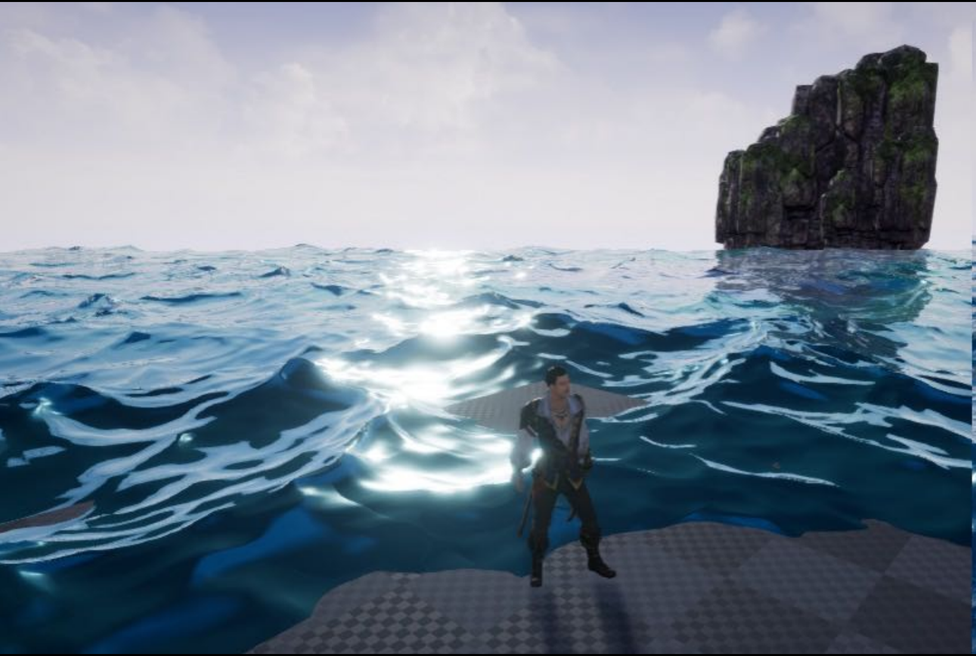
Ocean Material : Planar Reflection

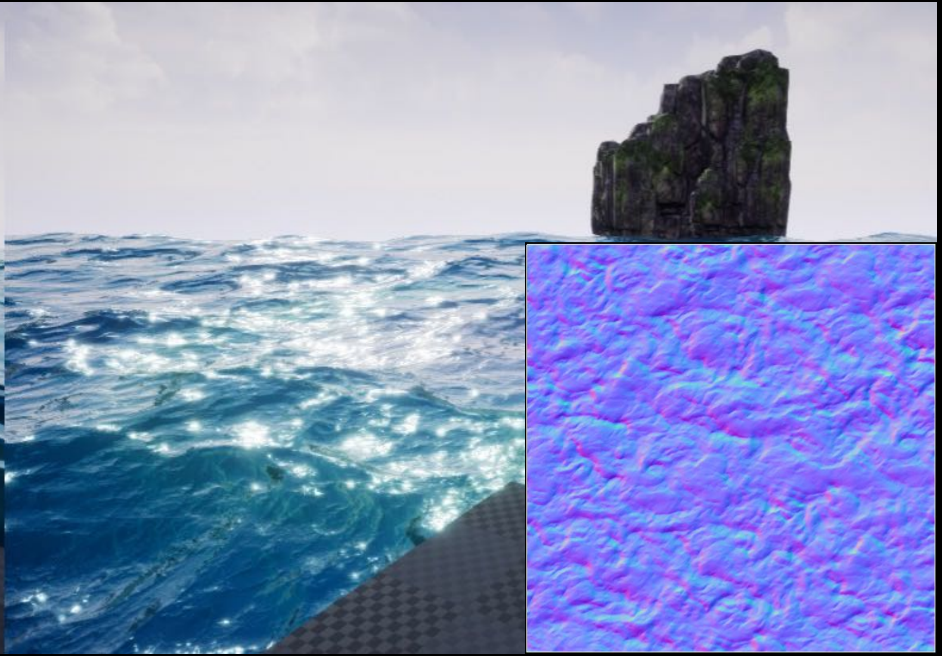
Ocean Material : Detail Normal



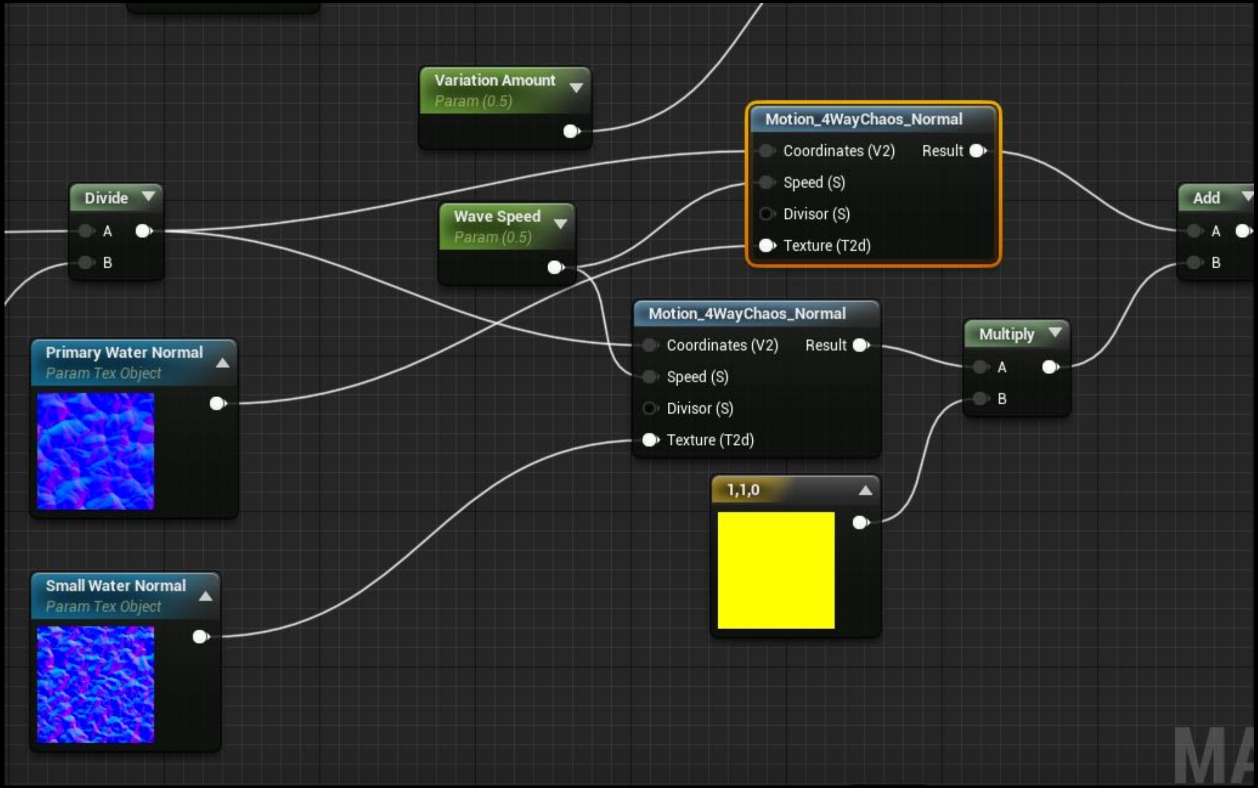


Ocean Material : Detail Normal

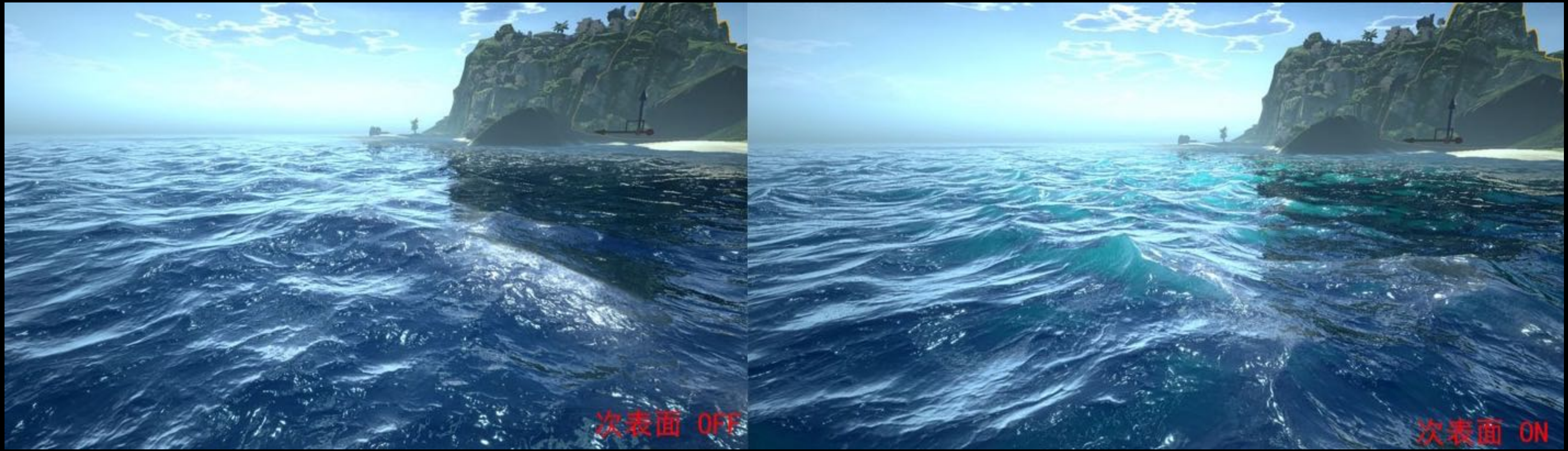




Ocean Material : Detail Normal



Ocean Material : Subsurface Scattering



Ocean Material : Subsurface Scattering

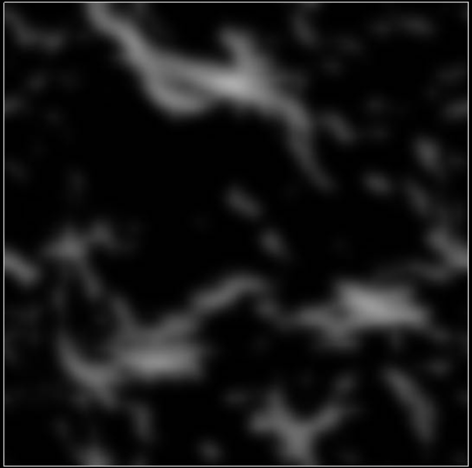
```
// Calculate each vertices' position offset
float3 SubsurfaceShadingSubsurface( FGBufferData GBuffer, float3 L, float3 V, half3 N )
{
    float3 SubsurfaceColor = ExtractSubsurfaceColor(GBuffer);
    float Opacity = GBuffer.CustomData.a;

    float3 H = normalize(V + L);

    // to get an effect when you see through the material
    // hard coded pow constant
    float InScatter = pow(saturate(dot(L, -V)), 12) * lerp(3, .1f, Opacity);
    // wrap around lighting, /(PI*2) to be energy consistent (hack do …
    // Opacity of 0 gives no normal dependent lighting, Opacity of 1 …
    float NormalContribution = saturate(dot(N, H) * Opacity + 1 - Opacity);
    float BackScatter = GBuffer.GBufferAO * NormalContribution / (PI * 2);

    // lerp to never exceed 1 (energy conserving)
    return SubsurfaceColor * lerp(BackScatter, 1, InScatter);
}
```

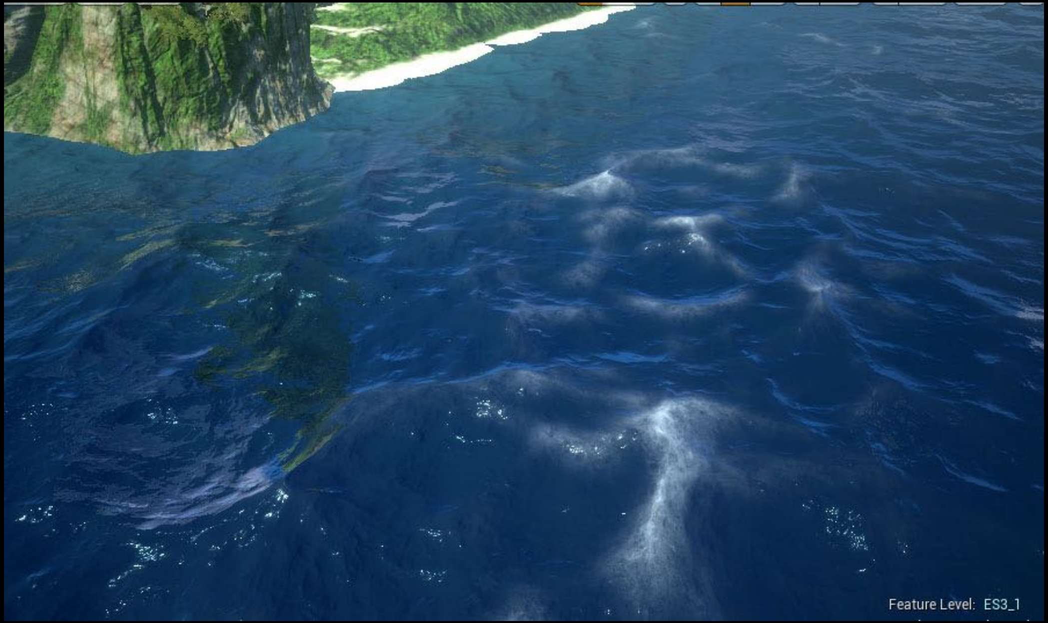
Ocean Material : Subsurface Scattering

Ocean Material : Subsurface Scattering

Ocean Material : Foam

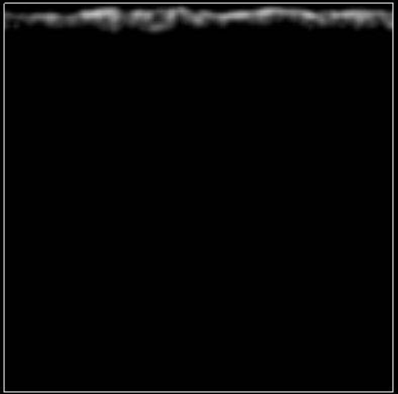
- Pre-generated foam masks
- Break the foam repetition rate with noise maps





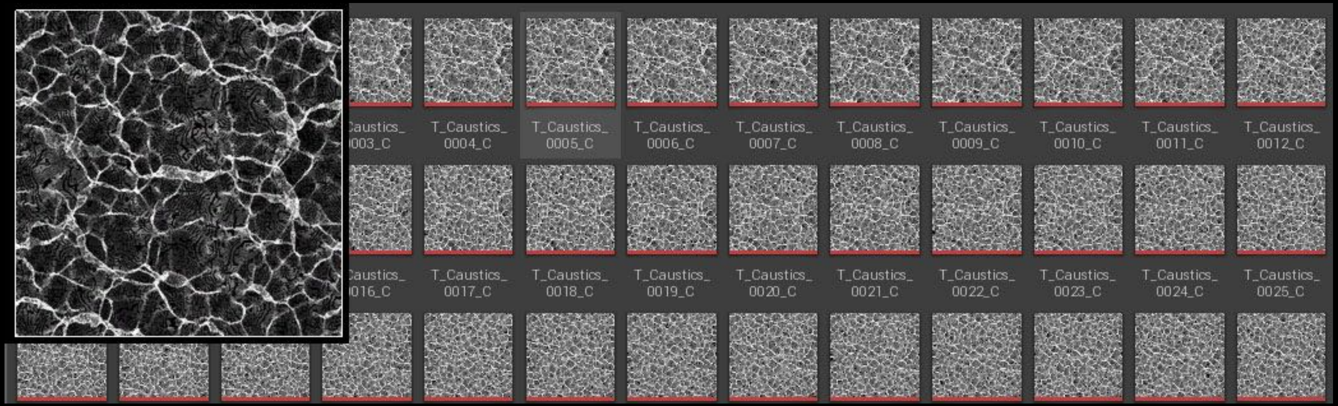
Ocean Material : Coastal Foam

- Shallow water is translucent
- Coastal foam use the depth of the coast(pre-baked maps), as the V coordinate to sample the coastal foam mask.



Ocean Material : Caustics

- The caustics algorithm comes from Epic Ryan Bucks GDC 2017 Talk : “Content-Driven Multipass Rendering in UE4 | GDC 2017 | Unreal Engine” (Caustics start at 18’21’’)
- Also use the Flipbook pre-calculation method.
- 64 frames of 256x256 textures



Implementation of Ocean System

Flipbook FFT Ocean implementation on mobiles

- Wave animation
- Ocean surface material
- **Sailing and water surface interaction**

Interaction with water: Buoyancy

- 4 sphere detection points around the hull
- Calculate buoyancy separately for each sphere. Convert the height difference into the rotational component of the ship.

Interaction with water: Buoyancy

- 4 sphere detection points around the hull
- Calculate buoyancy separately for each sphere. Convert the height difference into the rotational component of the ship.

UNREAL ENGINE

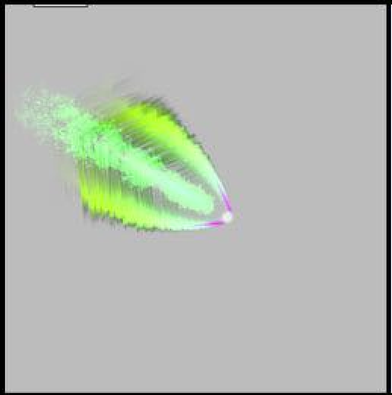
Interaction with water: Wake Wave

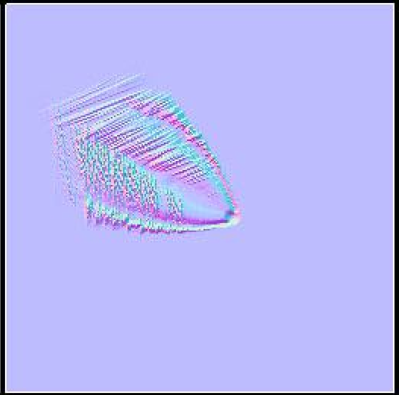
A faked fluid simulation, much more like a “Lake” wake wave effect.

Implementation can be roughly referred to the “BlueprintRenderToTarget” example in project “ContentExamples”

Interaction with water: Wake Wave

- Particle simulation. Using particle system to generate a wake wave height map.
- Use the height map as the displacement map for ocean surface.
- Generate the corresponding normal map from the height map, and apply for the ocean surface material.
- Easy for artist to control the visual effects.





Interaction with water: Wake Wave

- Particle Only, without ocean surface displacement.
- Cheap.

Mobile Device Optimizations

Full features only for high performance devices

Take one or more features below out for low performance devices:

- Subsurface Scatter
- Wake wave displacement (Use simple particle system)
- Planar reflection (Use Cubemap reflection instead)
- Detail normals

Mobile Device Optimizations

For the planar reflections, is also an expensive effect. It render all the objects in an extra render pass, that means it doubled the drawcalls and GPU time.

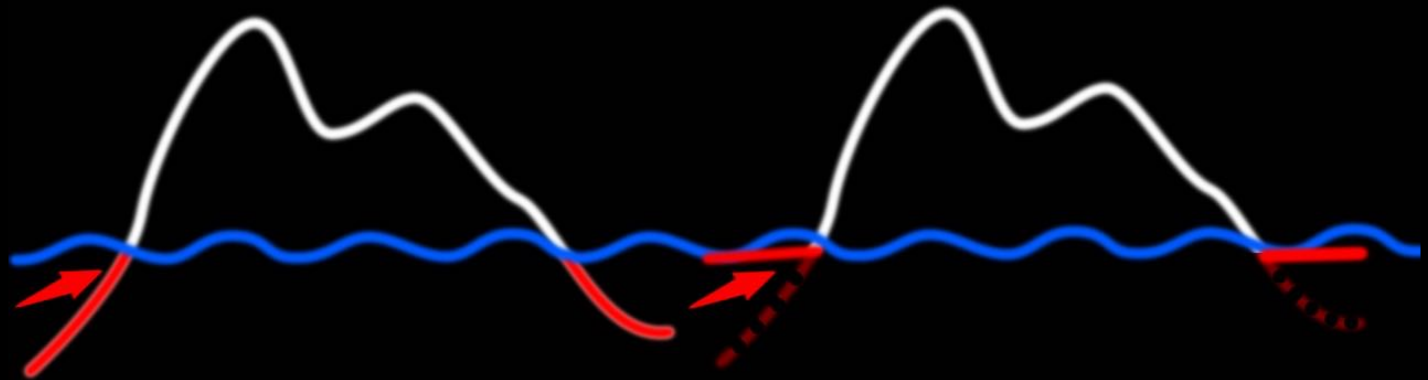
Optimizations:

- Make the planar reflection render target size as small as you can.
- Do not reflect all the actors, only the main part of them
- Use proxy mesh instead of static mesh actors if you have HLOD
- Disable unused effects in PR pass, such as PP, shadows depends on your situation.
- Do NOT enable global clip plane, it’s expensive for mobile devices.

Mobile Device Optimizations

For the global clip plane, if disable it, you will get an warning and a wrong reflection effect.

Here we use a trick to avoid this.



Some of the References

- GPU Gems 1, Ch1 Effective water simulation
- Simulating Ocean Water by Jerry Tessendorf
- Introduction to Algorithms, Ch 30 the FFT part
- Assassin's Creed III: The tech behind (or beneath) the action
- Content-Driven Multipass Rendering in UE4 | GDC 2017 by Ryan Bucks

Q&A

Thank you!