ULTIMATE WATER SYSTEM
V2.1.0

INTRODUCTION

ULTIMATE WATER is a successor of the PLAYWAY.WATER system, with many new features and improvements.

The documentation firstly focuses on setup guides, that show the basic concepts of the plugin.
Later, more in-depth overview of the each system and component is presented.

The plugin imports into *Ultimate Water System* folder and can be moved anywhere in your project.

> **important:** you can move the plugin folder anywhere in the project, but do not change the folder name

We strongly recommend to start by analyzing the sample scenes located at *Ultimate Water System/Samples/*.
They provide feature-by-feature overview of the plugin.

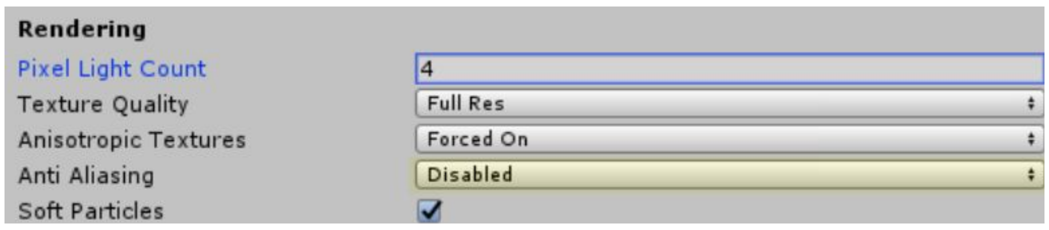
If you experience some problems, please check the Common Issues section.
All the known issues are tracked via github: https://github.com/Moonlit-Games/Ultimate-Water-System/

Support email: support@moonlit.pl
Unity forum thread:
https://forum.unity3d.com/threads/ultimate-water-system-the-most-advanced-realtime-water-simulation.493984/

GETTING STARTED (BASIC)

This section focuses on the simplest possible scene setup with Ultimate Water. The hyperlinks in headers and descriptions navigate to more in-depth documentation of specific component and/or feature.

important: for the water to work correctly, you must disable *Anti Aliasing* via *Edit/Project Settings/Quality* in the *Rendering* tab.



Rendering
Pixel Light Count 4
Texture Quality Full Res
Anisotropic Textures Forced On
Anti Aliasing Disabled
Soft Particles ☑

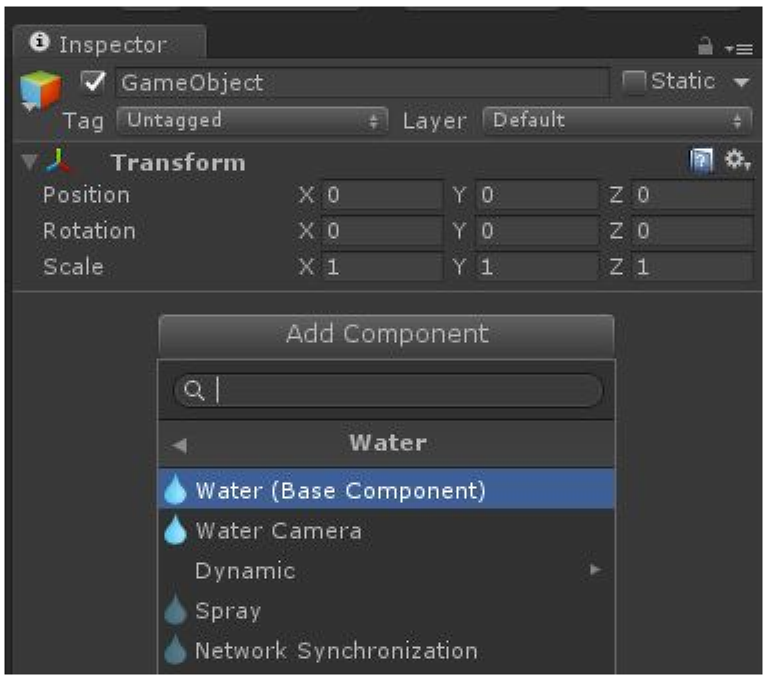
WATER COMPONENT

Create new *Water* in your scene by:

- Creating new gameobject, then adding (from the *Add Component* menu) *Ultimate Water/Water (Base Component)*,
- Right clicking in the hierarchy view and selecting *Ultimate Water/Water (Base Component)*

- You will be automatically prompted about adding it to all the cameras, but for the purpose of this introduction, we will add this component ourselves, click *cancel*.

Water Component is responsible for water rendering using selected settings.



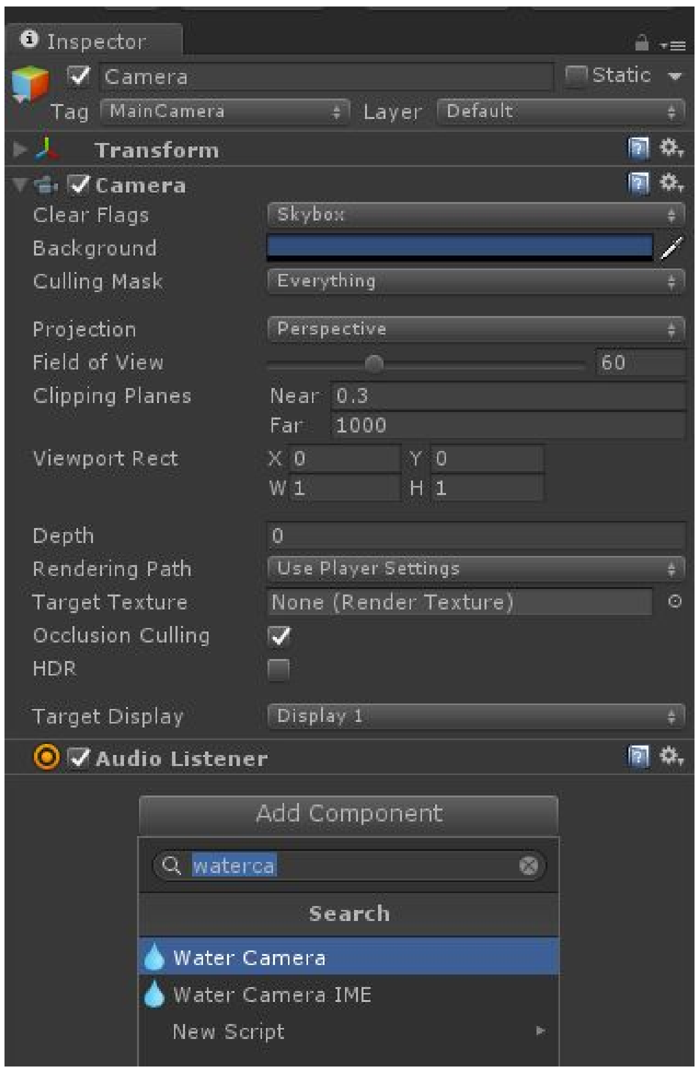
Inspector
☑ GameObject □ Static
Tag Untagged Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

WATER PROFILE

The *Profile* asset defines physical properties of the water (how the water looks, ie. diffuse color, index of refraction, …)
Each *Water* component can have different *Profile* assigned.
There are several default *Profiles* included (located in *Ultimate Water System/Samples/Profiles)*, you can use them as a base for your tweeks. Creating new profiles is described in *Water Profile Component Overview*.

WATER CAMERA

It's the final component that we need to start playing with our water. After selecting your *Main Camera*,
add the *Ultimate Water/WaterCamera* component to it.



Inspector
☑ Camera □ Static
Tag MainCamera Layer Default
Transform
☑ Camera
Clear Flags Skybox
Background
Culling Mask Everything
Projection Perspective
Field of View 60
Clipping Planes Near 0.3
Far 1000
Viewport Rect X 0 Y 0
W 1 H 1
Depth 0
Rendering Path Use Player Settings
Target Texture None (Render Texture)
Occlusion Culling ☑
HDR □
Target Display Display 1
☑ Audio Listener

THAT'S IT!

The basic setup is now complete, you can hit *Play* button to see the wavy, wavy waves

GETTING STARTED (Dynamic waves)

This part of the guide will walk you through the setup of the dynamic wave simulation.
Dynamic waves allow objects to interact with water, creating waves that collide with shore's, other objects, etc.

important: you should have completed Getting Started (basic), and have the water scene setup and ready

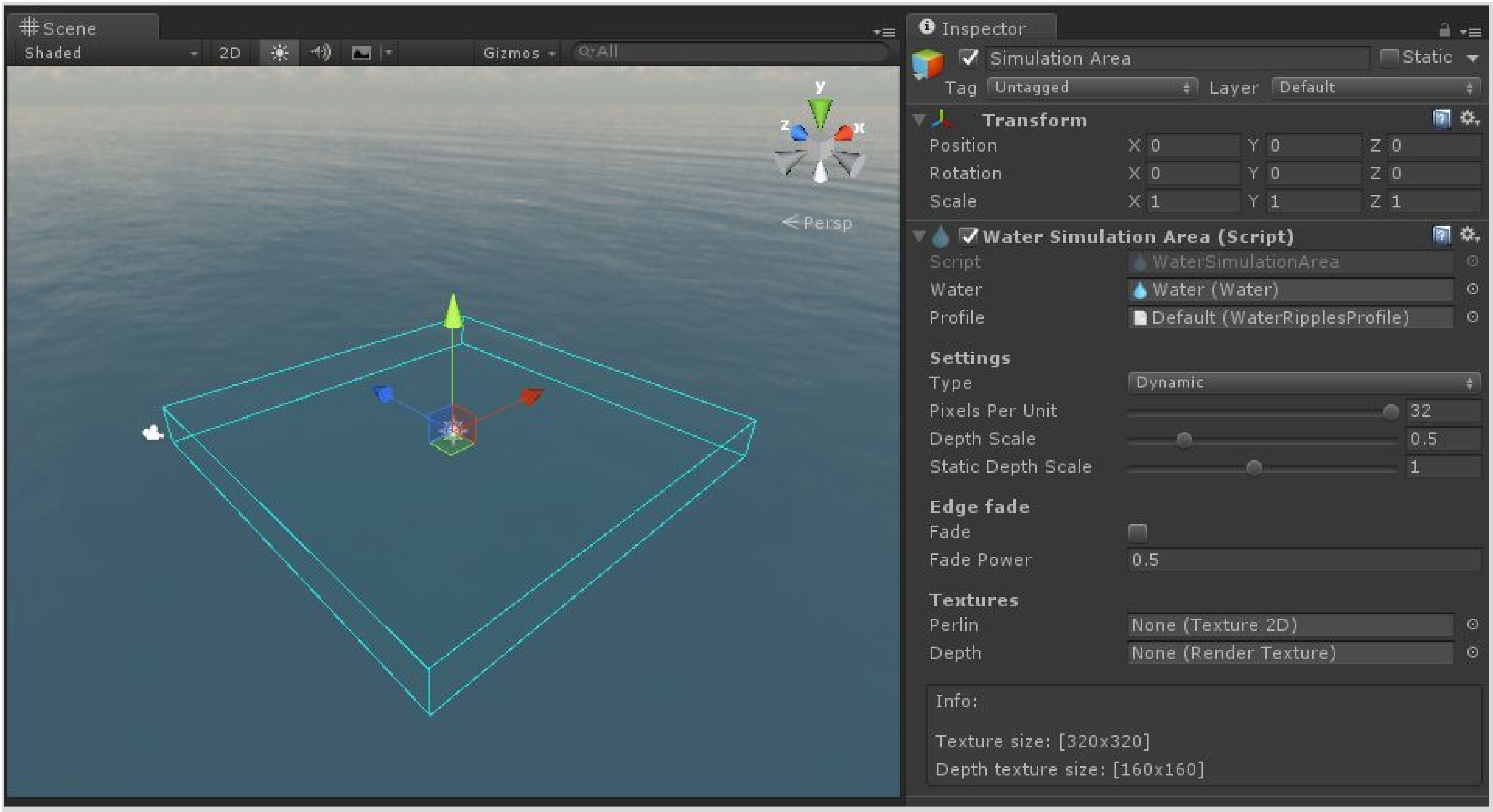
WATER SIMULATION AREA

We must define a space, where the simulation will happen, we can do this by adding *Water Simulation Area*:

- creating new *GameObject* and adding *Water Simulation Area* component or
- right clicking in the hierarchy and choosing *Ultimate Water/Simulation Area*

We can see that the cyan-colored gizmo shows up, it represents the area in which the simulation will occur.
In the *Inspector* settings of this component we can set the reference to the *Water* object,
that we want our simulation to interact with.

note: if there is only one *Water* object on scene, *WaterSimulationArea* will automatically assign it to itself



☑ Simulation Area □ Static
Tag Untagged Layer Default

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

☑ Water Simulation Area (Script)
Script WaterSimulationArea
Water Water (Water)
Profile Default (WaterRipplesProfile)

Settings
Type Dynamic
Pixels Per Unit 32
Depth Scale 0.5
Static Depth Scale 1

Edge fade
Fade □
Fade Power 0.5

Textures
Perlin None (Texture 2D)
Depth None (Render Texture)

Info:

Texture size: [320x320]
Depth texture size: [160x160]

WATER RIPPLES PROFILE

The *WaterRipplesProfile* is analogous to *WaterProfile*, and defines settings for generating dynamic waves.

- Right click inside *Project View*
- Select *Create/Ultimate Water/Ripples Profile*

INTERACTIVE OBJECT

Interactive objects interact with *Simulation Areas*.

- Create simple sphere gameobject by selecting *Create/3D Object/Sphere* in *Hierarchy* context menu
- *Add WaterInteractive* component

THAT'S IT!

Now, hit the *Play* button and move the sphere around (in editor) to see dynamic waves it generates.

GETTING STARTED (underwater effects)

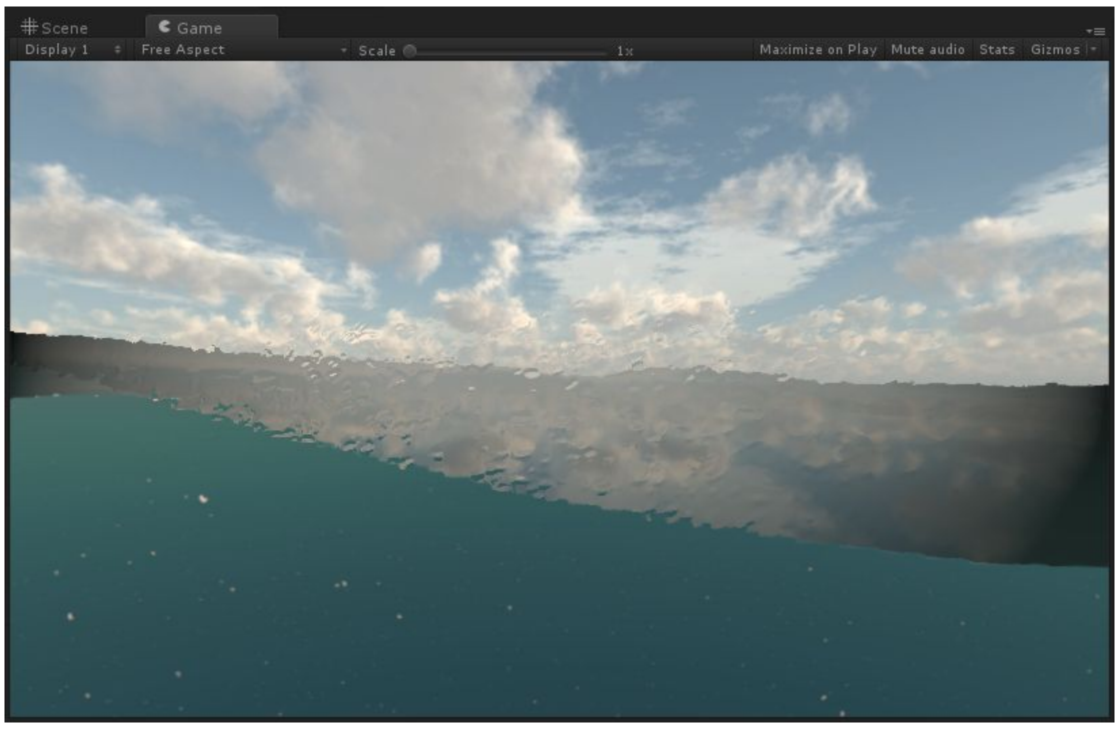
This section shows, how to configure your *Camera* to enable underwater effects.

important: you should have completed Getting Started (basic), and have the water scene setup and ready

UNDERWATER IME

Handles underwater rendering. The colors and other underwater settings are modifiable through *Water Profile*.

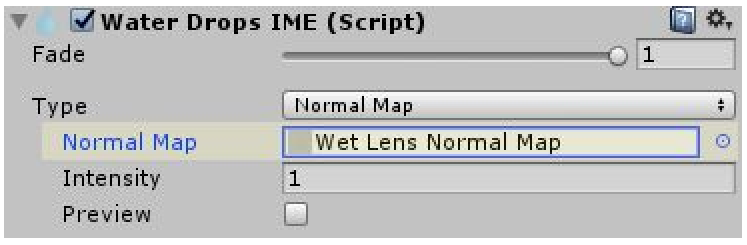
- Select your water-rendering *Camera*,
- Click *Add Component* and type "*Underwater IME*" or select the component from *Ultimate Water/UnderwaterIME*



WATER DROPS IME

Applies camera distortions after emerging from water.

- Make sure that you're *not* in playmode
- Select *Normal Map* type
- Choose texture, you can use provided *Wet Lens Normal Map*
- Modify *Intensity* to change the lens distortion strength



☑ **Water Drops IME (Script)**

Fade 1

Type Normal Map

Normal Map Wet Lens Normal Map

Intensity 1

Preview □

THAT'S IT!

Now, submerge your Camera in water!

GETTING STARTED (PHYSICS)

Adding physics to your objects allows them to interact with water (float, drift).

important: you should have completed Getting Started (basic), and have the water scene setup and ready

WATER PHYSICS

Performs physically-correct calculations of *Rigidbody* behaviour in water.

- Create *Cube GameObject*
- Add *Water Physics* component either by clicking *Add Component* and typing *"Water Physics"* or selecting *UltimateWater/Physics*
- Decrease *Buoyancy Intensity* or increase *Mass* of the *Rigidbody* until the *Gravity Balance* (shown at the bottom of the *Water Physics* scripts) will be around 100%.

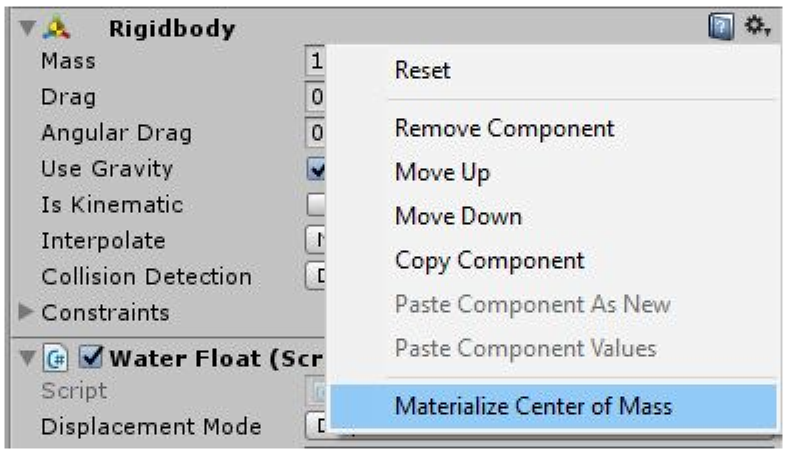
note: *Gravity Balance* of 100% means, that the object is in equilibrium, anything below that and the object drowns, anything above and the object will surface eventually.

Center of mass:

By default *Center of Mass* of the object is assumed to be in the geometrical center, that is not the case for some objects (like ships for example, where the center of mass is very low).

To modify the *Center of Mass* you can:

- Click on the cog ⚙ of the *Rigidbody* component
- Select *Materialize Center of Mass*
- New child object will be created with the name *Center of Mass*, that you can move it to desired position



WATER FLOAT

Simplified version, that binds object to the current water point. It can be used for small objects (debris, etc.).

- In your *Water* component, select *Profile - Moderate*, it's best suited for floating objects demo
- Create *Sphere GameObject*
- Add *Water Float* script through *Add Component* menu
- Assign the *Water* reference (if it wasn't assigned automatically)

THAT'S IT!

Press *Play* and watch your cube floating on water.

GETTING STARTED (RESTRICTED WATER)

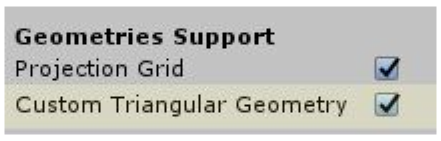
The standard water setup show how to create infinite water, this section describes necessary steps to restrict the water area in various ways.

important: you should have completed Getting Started (basic), and have the water scene setup and ready

CUSTOM MESHES

Defines the surface used for *Water* rendering.
Firstly, we need to check if our *Shader Set* supports *Custom Meshes*

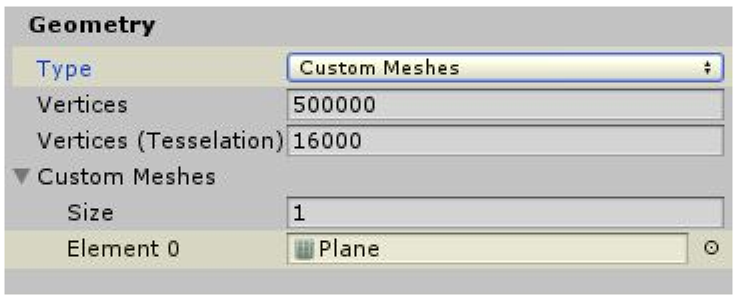
- Select *Water component*
- Click *Edit* on the selected *Shader Set*
- Ensure that in the *Geometry Support* tab, the *Custom Triangular Geometry* is enabled
- If not, enable it and click *Apply Changes* located at the bottom of the inspector



Geometries Support
Projection Grid ☑
Custom Triangular Geometry ☑

Secondly, we must set up our *Water* component.

- Select *Water* component
- Change *Geometry Type* to *Custom Meshes*
- Add mesh to *Custom Meshes* list
  (in this example we'll use standard *Plane* mesh)



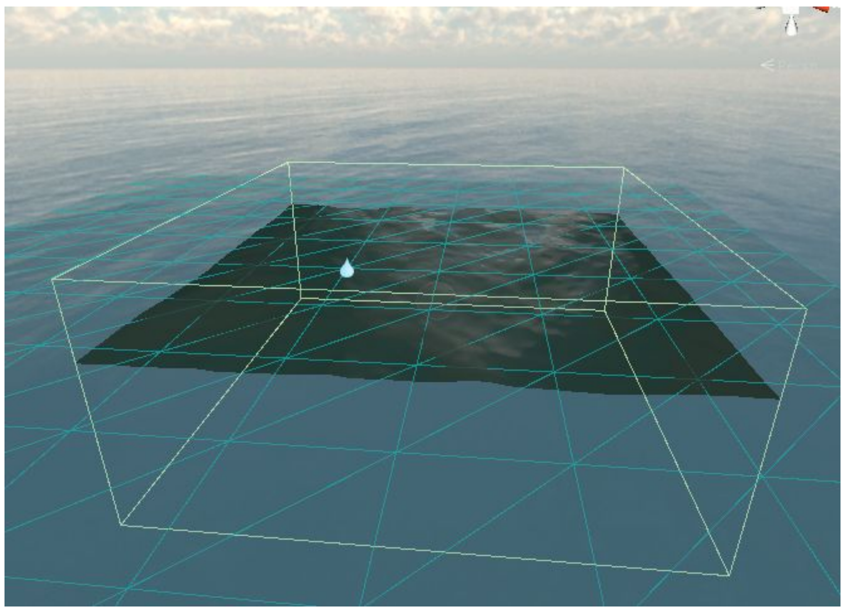
Geometry
Type: Custom Meshes
Vertices: 500000
Vertices (Tesselation): 16000
▼ Custom Meshes
Size: 1
Element 0: Plane

note: added custom meshes are internally used to render water, where the mesh would normally be drawn, this in in respect to *Water* gameobject position.

WATER VOLUME ADD

Defines the space, to which the water should be restricted to.

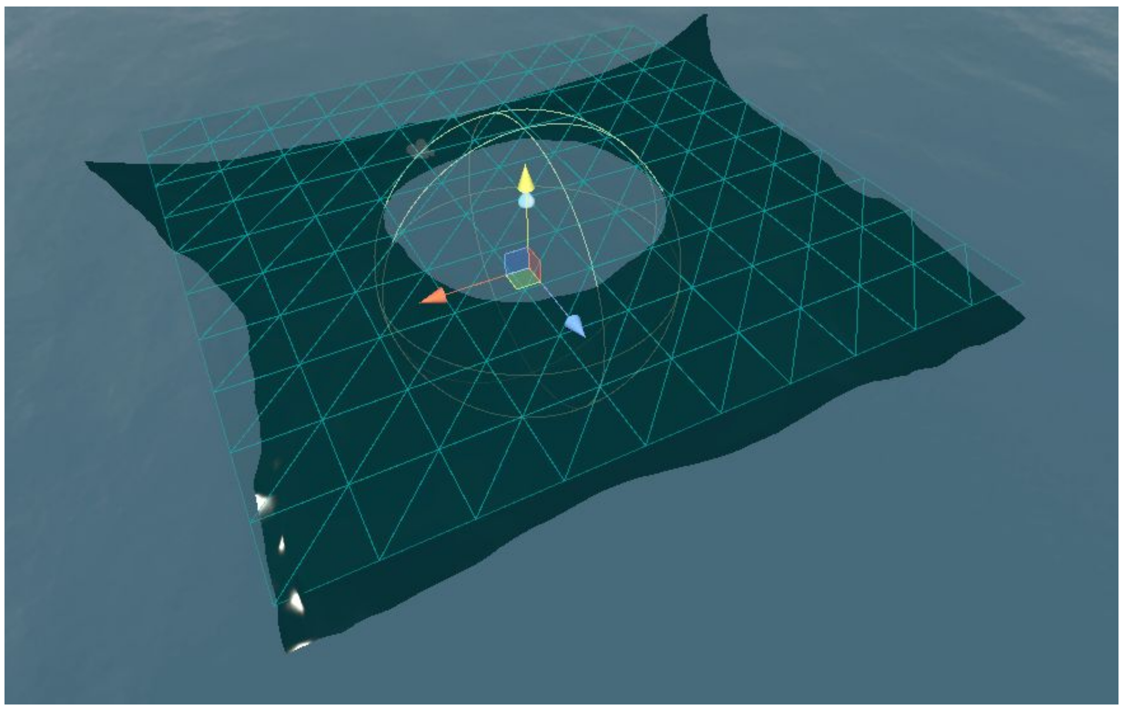
- Create empty *GameObject*
- Add *Water Volume Add* script
- Add *Collider* component, that defines the space, in which the water should be rendered
- Assign the reference to *Water* component (if it was not assigned automatically)



WATER VOLUME SUBTRACT

Removes water in the volume specified by attached collider.

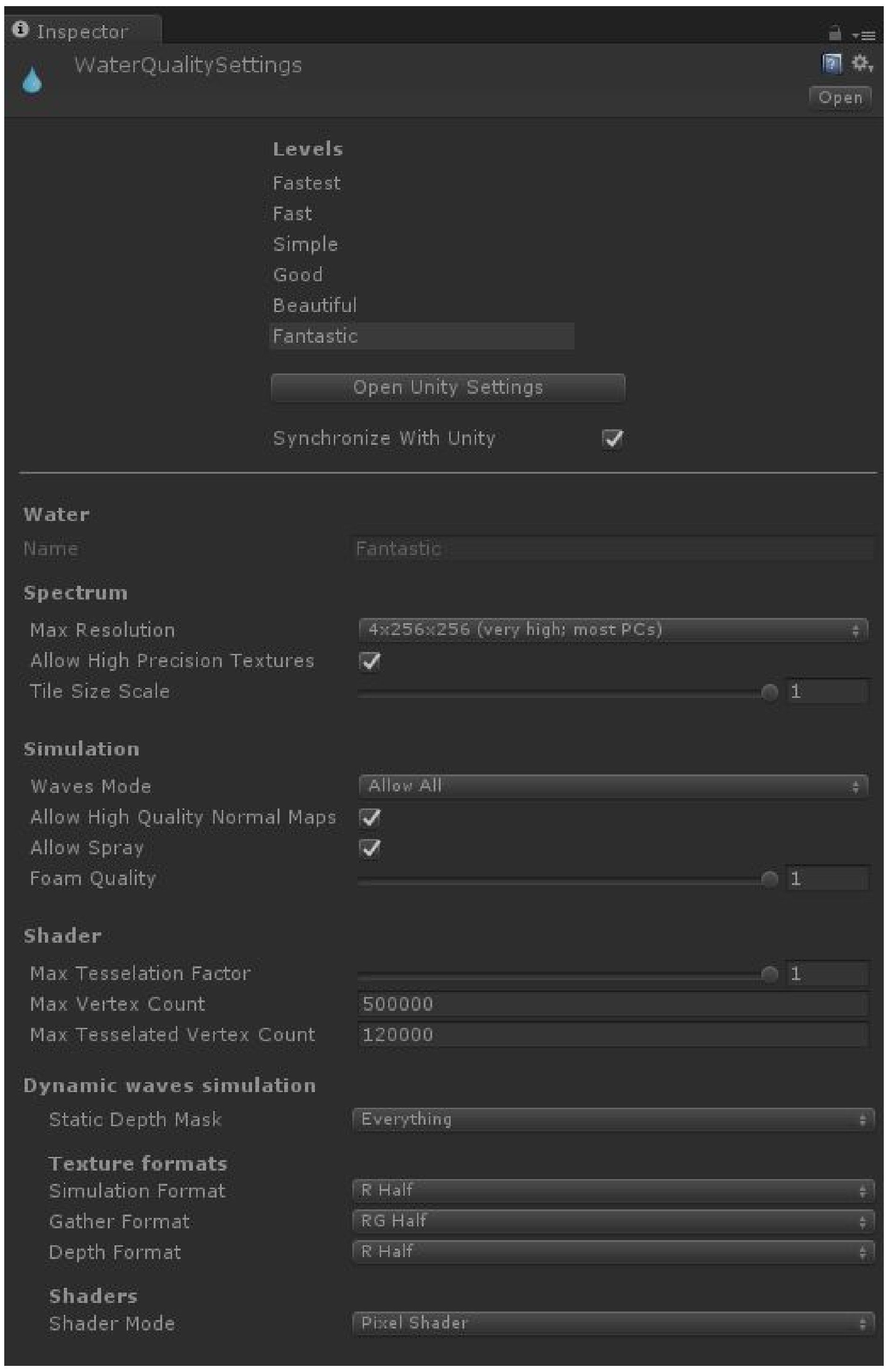
- Create empty *GameObject*
- Add *Sphere Collider*
- Add *Water Volume Subtract*



COMPONENT OVERVIEW

GLOBAL QUALITY SETTINGS

You can access global quality settings by selecting *Edit/Project Settings/Water Quality*.
They are organized just like the Unity quality settings. You can choose between different quality levels and for each of them customize performance-related settings. They contain water rendering settings as well as dynamic waves simulation settings.
By default the names, order and current quality level selection are synchronized with Unity settings.
This can be disabled with *Synchronize With Unity* toggle.



Levels
Fastest
Fast
Simple
Good
Beautiful
Fantastic

Synchronize With Unity ☑

Water
Name: Fantastic

Spectrum
Max Resolution: 4x256x256 (very high; most PCs)
Allow High Precision Textures ☑
Tile Size Scale: 1

Simulation
Waves Mode: Allow All
Allow High Quality Normal Maps ☑
Allow Spray ☑
Foam Quality: 1

Shader
Max Tesselation Factor: 1
Max Vertex Count: 500000
Max Tesselated Vertex Count: 120000

Dynamic waves simulation
Static Depth Mask: Everything

Texture formats
Simulation Format: R Half
Gather Format: RG Half
Depth Format: R Half

Shaders
Shader Mode: Pixel Shader

Spectrum:

Max Resolution - [performance-critical] all non-interactive waves simulations will be performed at most with this resolution.
Allow High Precision Textures - determines if high precision textures (32 bit floating point per channel) may be used, if it's supported by the target hardware. This setting is viable only for extreme setups with simulation resolution 1024x1024 and higher.
Tile Size Scale - allows you to globally scale the tile size of all water simulations.
Setting it below 1.0 will increase quality and compensate drawbacks of using small simulation resolution, but will make repeating pattern of water more noticeable.

Simulation:

Waves Mode - determines the allowed simulation types:

- Allow All - both simulation types (FFT and Gersner) are allowed,
- Allow Slope FFT - Gernser waves (low-quality) are allowed and used as a fallback to FFT. FFT may be used only to generate high-quality water normal maps,
- Allow Gernser - only Gernser waves (low-quality) are allowed,
- Disallow All - no waves will be produced.

Allow High Quality Normal Maps - analogous to *precision textures*, it provides small quality improvement at a great cost. You should consider enabling it only in the highest quality level.

Shader:

Max Tessellation Factor - clamps tessellation factor.
Max Vertex Count - clamps vertex count.
Max Tessellated Vertex Count - clamps vertex count after tessellation.

Dynamic waves (water ripples) simulation:

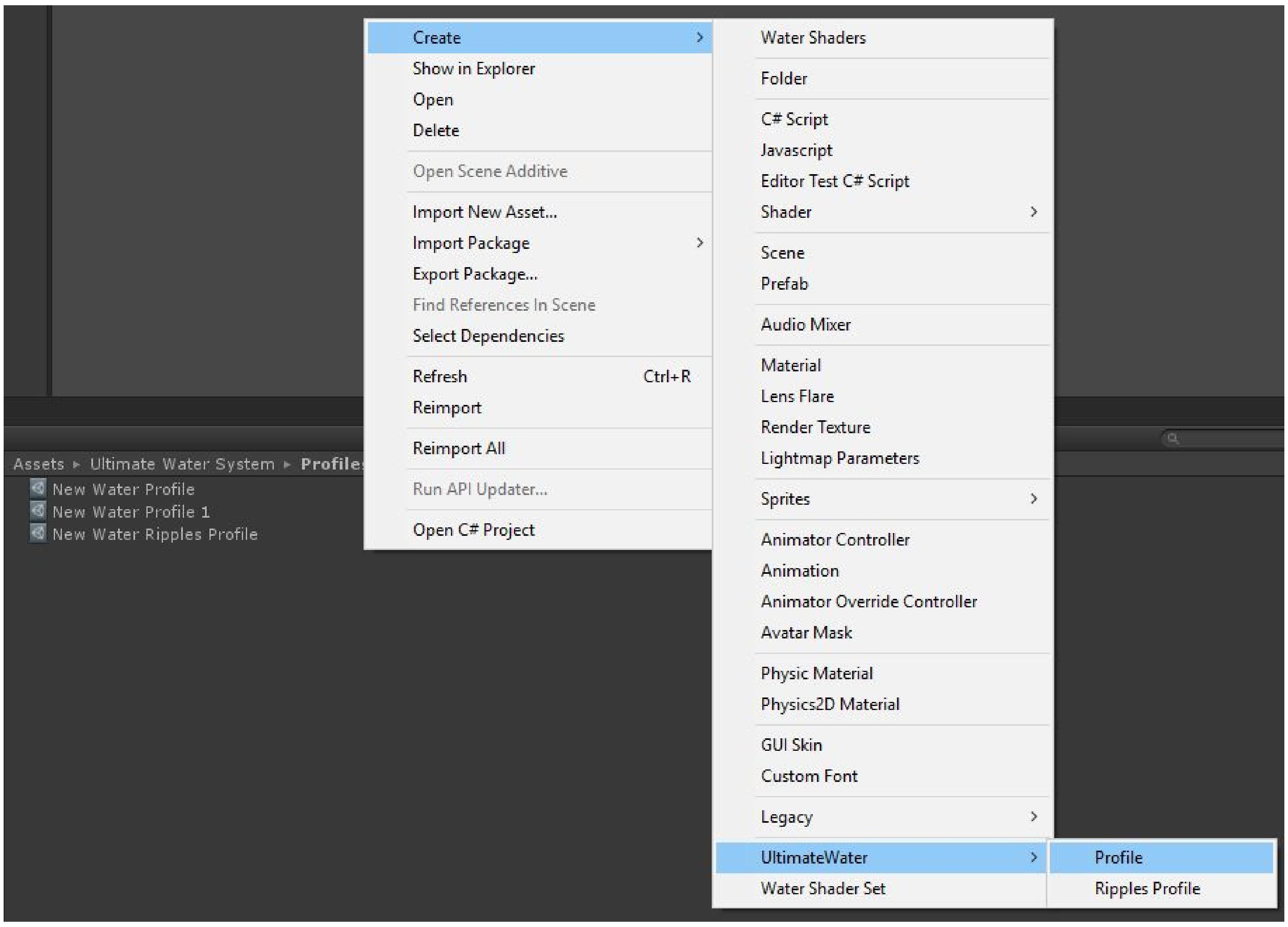
Static Depth Mask - [performance-critical] determines, which objects will be interacting (stopping, reflecting) with dynamic waves. This should only include necessary objects as it can result in rendering them each frame. It can be disabled altogether, as its effects are often subtle.
Iterations - [performance-critical] speeds up wave propagation, by performing more per-frame (Fixed Update) computation steps.
Texture Formats - [advanced] how precise the calculations will be, the half-precision format (16 bits per channel) is enough for most, if not all purposes. If you have too many gpu-cycles to spare and don't know what to do with them, you can use floating point textures here.
Shaders - [advanced] dynamic waves are implemented both in pixel and compute shader, the pixel shader should be faster but, if you can, test it on the target hardware and compare the performance on case-by-case basis.

WATER PROFILE

Defines physical properties of the water as well simulation parameters. The parameters related to spectrum generation (Spectrum Type, Wind Speed, Tile Size, Waves Amplitude and Frequency) cannot be changed at runtime, because of the high computational cost. For dynamic changes in the weather conditions, blending between two or more different profiles should be used, this is shown in *Ultimate Water System/Samples/Profile Blend* demo scene.

Setup:

- Right click in the *Project View*
- Select *Create/Ultimate Water/Profile*
- Return to your *Water* component and select newly created profile from *Water/Profile* dropdown



Parameters:

Spectrum Type - two spectrum types are currently implemented: Phillips (legacy) and Unified (recommended).
Wind Speed - wind speed in knots. Hint on the left will display its Beaufort classification.
Tile Size - determines area of water simulation in real world units. Use low values (180 or lower) if your cameras won’t get higher than 20 meters above the water surface (FPS games etc.). Try higher values if your cameras operate naturally on heights above the 20 meters (RTS games) to reduce tiling.
Waves Amplitude - too big wave amplitude in proportion to wave frequency results in clearly visible wave artifacts (waves going through itselfs).
Waves Frequency Scale - increases wave count, should be tuned along with amplitude.
Horizontal Displacement Scale - how far the waves move horizontally.
Directionality - eliminates waves moving against the wind. Should be 0 for oceans.
Fetch - it is the length of water in meters over which a wind has blown. Usually a distance to the closest land in the direction opposite to the wind.

Colors:

Diffuse - diffuse color of the water.
Reflection - the color of the reflection.
Absorption Edit Mode - allows you to choose between absorption modes:

- Transmission - [recommended] how much light passes per meter of water (it's easier to setup),
- Absorption - how much light of each color (R, G and B) should be absorbed per meter by the water volume.

Custom Underwater Transmission - changes the transmission characteristics when underwater.
Index of Refraction - refer to the sources around the web to find the correct value for the fluid you need. Should be set to 1.33333 for physically correct water.

Subsurface Scattering:

How much light should be scattered back from the underwater. Should be set high in profiles with small and medium waves, low in profiles with big waves.

Isotropic - non-directional scattering, the light is scattered in all the directions equally.
Forward - directional, light is scattered more in the light emission vector.
Contrast - enhances computed scattering visibility.
Directional Wrap SSS - crude approximation of subsurface scattering. Use it along with with the full volumetric scattering defined above to get the best results. This setting is used for directional lights.
Point Wrap SSS - same setting, but for point lights.

Basic Properties:

Smoothness - analogous to the *Smoothness* in *Standard Shader*.
Custom Ambient Smoothness - override for ambient/skybox reflections.
Dynamic Smoothness Intensity - change in smoothness over distance.
Refraction Distortion - determines refraction distortions intensity.
Edge Blend Factor - controls water fading near the intersections with other geometry.
Density - physical density (for objects using *Water Physics*). Dense fluids apply stronger forces to objects moving through them.

Normals:

Fade Distance and Bias - water gets really noisy at a distance and SMAA or FXAA won't handle that. This parameters will let you fade water's normals to avoid this problem.
Slope Intensity - intensity of wave normal when using FFT.
Tiles - define parameters of two normal map tiles used by the shader.

Foam:

Waves generate foam through their *Horizontal Displacement*, so it must be non-0 for the foam to work

Intensity - How much foam the waves generate.
Threshold - The bigger the threshold, the smaller the waves need to be to generate foam.
Fade Factor - Determines how fast the foam disappears.
Foma Shore Intensity - How
Foam Normal Scale - Multiplier for the *Foam Normal Map*.
Foam Diffuse Color - The color of the foam.
Foam Specular Color - Specular color of the foam, reflection in alpha channel.

Planar Reflections:

Intensity - how visible the reflections are.
Flatten - flattens mapping of the reflections to the water surface.
Offset - vertical offset of the planar reflection. Fixes some common artifacts produced by them.

Underwater:

Those parameter are used by *Underwater IME* script.

Blur Size - How much blur will be applied when underwater.
Underwater Light Fade Scale - Light absorption by depth. [0 - no absorption]
Distortion Intensity - How much distortion is applied.
Distortion Animated Speed - the speed of the distortion changes.

WATER

The component responsible for water rendering. By default, water covers whole scene (like an ocean).
You can restrict water areas and create more than one water objects on scene. Creating restricted water is covered in Getting Started (Restricted water).

Parameters:

Profile - allows you to choose between created *WaterProfiles*.
Shader Set - which shader set should water use, shader set defines enabled features of water rendering.
Synchronize - if disabled, you can modify (at runtime) *WaterProfileData* independently of the profile stored in the asset. Leaving it enabled assures that any changes to the profile will be propagated to renderer. Enabling this always causes water to synchronize with the *Profile Asset*.

Environment:

Reflection Probe Anchor - defines probe anchor for water rendering.
Boundless - should the water be considered endless in all directions? Used by physics etc. to determine where the water is. If you won't check it, you should create game objects with colliders and *WaterVolumeAdd* component to specify water volume for proper physics and underwater effect detection.
Seed - setting it to 0 makes water behave differently at each run. Other values make it predictable. It is useful for cutscenes, network games etc.

Geometry:

Vertices (Tesselation) - same setting, but used for systems with tessellation support. You want it to be lower as tessellation puts lots of well distributed additional vertices.

Shading:

Tesselation Factor - how many vertices should be added by tessellation. It is recommended to set it to 1.0 and control this by global water quality settings.
Type - [advanced] possible values are *Radial Grid*, *Projection Grid*, *Uniform Grid* and *Custom Meshes*. *Radial Grid* is the recommended geometry type. It is flat radial grid placed on the sea level, covering field of view as closely as possible. *Projection Grid* may be a good choice for platforms not supporting tessellation but creates noticeable artifacts at horizon. Due to having many drawbacks, this method may be removed in upcoming versions. *Uniform Grid* is used by some internal effects and not recommended in regular use. *Custom Meshes* is the geometry of choice if you plan to create small fluids like puddles. It gives you exact control over vertex count and surface area of fluid.
Vertices - determines vertex count of water if not using custom mesh.

Planar Reflections:

Reflection Mask - which layers should get reflected.
Reflect Skybox - toggles skybox rendering.
Receive Shadows - if checked, water will receive shadows from main directional light. That light needs to have *WaterShadowCastingLight* component attached to it. If water is opaque (blend edges off, refraction off) shadow receiving will work for all lights and no additional steps are necessary.
Resolution - [performance-critical] the resolution, at which the reflections are rendered, you should make sure that this parameter is as low as the final effect is satisfying.
Retina Resolution - [performance-critical] parameter analogous to the above mentioned, for high dpi displays.

Subsurface Scattering:

Mode - allows disabling the scattering.
Ambient Resolution - the resolution at which the calculations will be performed.
Light Layer - set to the layer of the lights.
Light Count - how many lights should be supported.

Wind Waves:

Resolution - [performance-critical] determines the quality of the waves simulation.
Desired Standard Error (CPU) - precision of water height sampling (in Unity units).
High Precision - as described in *Global Quality Settings*, default off.

Wind Direction Pointer - transform component that defines the wind direction with its Z-axis.
Loop Duration - leaving the duration at 0 ensures best quality, it can be set for optimising purposes (good values ~10 seconds)
Copy From - [performance-critical] sets wave spectrum from other water, making this instance alot faster. It's not recommended to use more than 1-3 unique spectrums.

Inspect - You may choose most of the water maps here and observe how their contents change in time while in playmode.

WATER CAMERA

Renders water on camera along with all other screen-space data. Water camera determines it's submersion state using *WaterSample*s, basic setup uses just one sample per camera with big threshold radius; if more precise behaviour is needed, you can increase subdivisions and decrease the radius of the spheres (show as gizmos in scene view).
note: camera is considered submerged if all the spheres are below the water surface

Parameters:

Water Geometry - [advanced] allows you to override water geometry used by the water to render itself. Useful for internal effects which will not work with specific geometry types.
Render Water Depth - [recommended] renders water depth into Unity's native depth map.
Render Volumes - renders data used by subtractive volumes.
Shared Command Buffers - [advanced] if you use other command buffers, you should enable this options.
Base Effect Quality - water has a pretty smooth shape so it's often safe to render its depth in a lower resolution that the rest of the scene. Although the default value is 1.0, you may try to set is as low as 0.5 and gain minor performance boost. If in the process, any artifacts become visible, reset it to default value.

Submersion (*state*) - shows the current camera state (above water, partially or fully submerged).
Subdivisions - how many sampling points are used to determine camera submersion state.
Radius - how big are the threshold spheres
Submersion State Changed (event) - invoked when camera changes above mentioned state.

Toggle SceneView rendering (button) - enables/disables water rendering in the scene view.

VOLUMES - ADD

Adds water to the insides of attached colliders. This is one way to specify where your water is, if you won't make it boundless. Use it to create a glass of water, a puddle etc.

note: make sure to enable the Boundless property in *Water component*
note: consider using Custom Mesh in *Water component* as it provides best performance for additive waters

demo: demo scene presenting this feature can be found at *Ultimate Water System/Samples/Physics*

Parameters:

Water - reference to the water component that should be affected.
Render Mode - determines the way water is removed
- None - disables water subtracting,
- Basic - "cuts" a hole in the water using water-collider cross section,
- Full - renders the water around the collider.

Affect Physics - if enabled, disables water physics calculations.

VOLUMES - SUBTRACT

Removes water from the insides of the attached colliders. It affects both rendering and physics.
Possible uses include boats, submarines, closed bottles, etc.

demo: demo scene presenting this feature can be found at *Ultimate Water System/Samples/Volumes*

Parameters:

The parameters are analogous to the Add version (mentioned above).
The only difference is, that the Basic render mode renders only surface of the water.

WATER PHYSICS

Simulates drag and buoyancy forces applied by the fluids on the attached object.

note: you can get a feeling how the object will behave by checking the computed *Gravity Balance* value.
The value of 100% means that the forces are equal, anything below that and the object will eventually sink.

demo: demo scene presenting this feature can be found at *Ultimate Water System/Samples/Physics*

Parameters:

Sample Count - how many sample points should be used to apply forces.
Drag Coefficient - determines how much drag forces should be applied.
Precision - for objects bigger than most of the waves, you should keep this parameter low, higher precision should be used for smaller objects, when you can observe erroneous behaviour.
Buoyancy Intensity - scales buoyancy, it can be useful for extremely low density objects (think beach balls) as the PhysX time step may be too high to handle them correctly.
Flow Intensity - scales the flow force applied by water waves, causing objects to drift.

WATER FLOAT

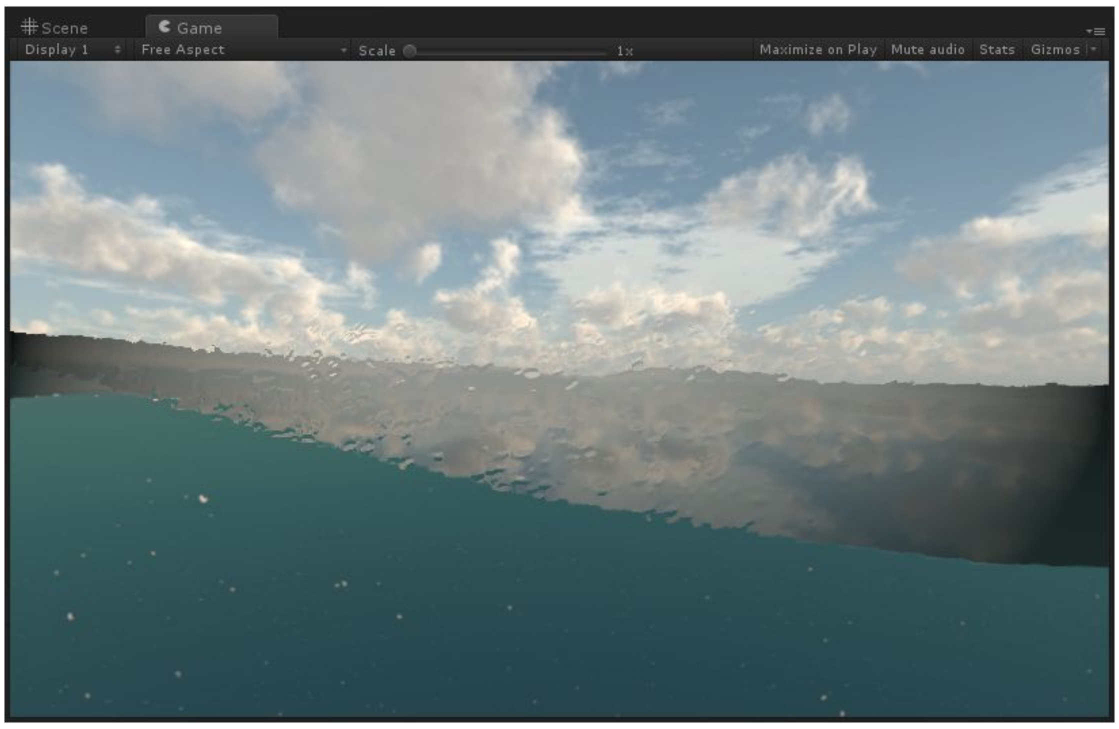
Makes objects float on water without the use of physics.

Parameters:

Height Bonus - vertical offset to the object position.
Precision - for objects bigger than the most waves, you should keep this parameter low. Higher precision should be used for smaller objects.
Displacement mode - determines the motion of the object:

- Displacement - object will follow the original point on water in all directions.
- Height - object will move only vertically. This mode can cause some imprecisions.

UNDERWATER IME



Adding *Underwater IME* to the *Water Camera* gameobject enables rendering the environment with camera underwater.

demo: demo scene presenting this feature can be found at *Ultimate Water System/Samples/Underwater Effects*

When using *Underwater IME* with water contained in *Additive* and *Subtractive Volumes* make sure that the *Boundless* option in *Water Component* is turned off.

Parameters:

Blur Quality - the iterations of the blur algorithm, should be kept as low as possible.
Underwater Audio - applies audio reverb filter when underwater.
Camera Blur Scale - per-camera blur scale, should only be used if per-camera changes in blur are needed, otherwise the *Water Profile* settings should be used.
Mask Resolution - when camera is partially under water, this resolution is used to draw a line between air and water.

note: with some settings, color banding may be visible, in that case, the easiest solution is to apply dithering, Unity provides free, *Post Processing Stack* package with one of the options being *Dithering post process.*

WATER DROPS IME

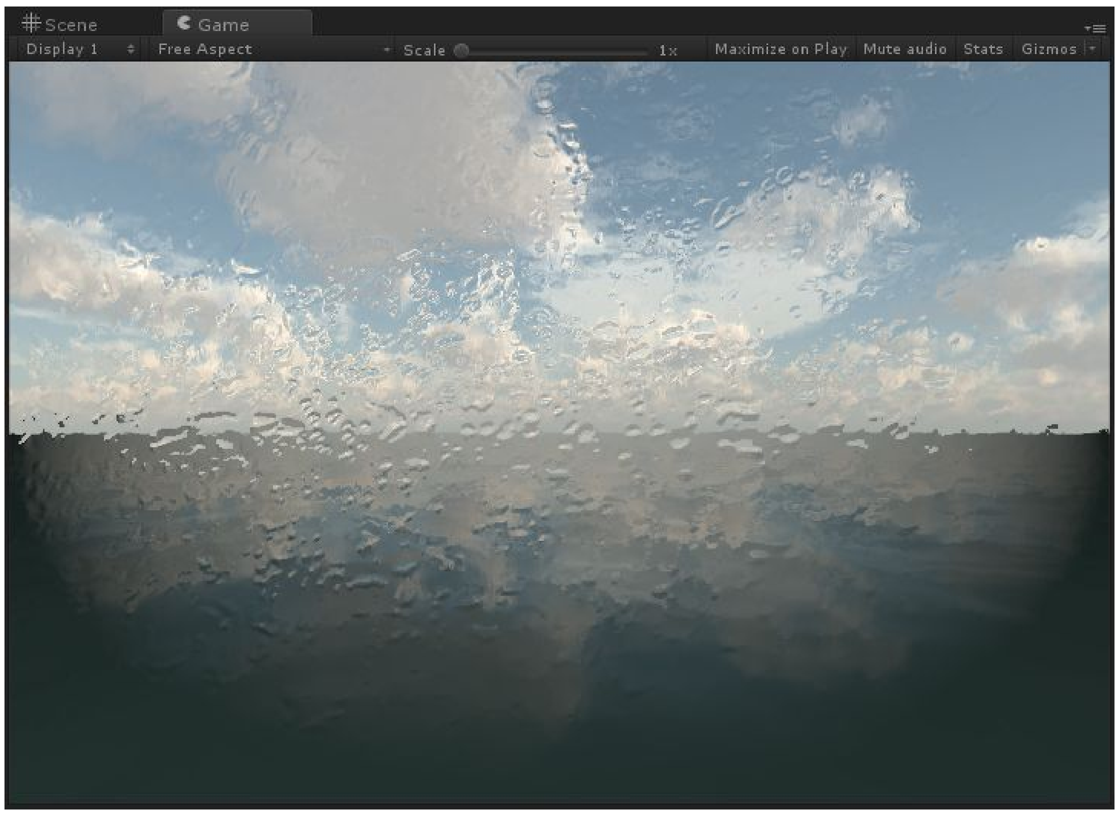
Add this component to your *Water Camera*. Used with *Underwater IME*. When camera emerges from water, *Water Drops* allows for water effects to appear on camera lens.

Parameters:

Fade - how fast the effect disappears.
Type - what effect will be applied when surfacing:

- Normal Map - creates distortions based on normal map texture provided.
- Blur - blurs the screen, more accurate with respect to how the human vision behaves.

Normal Map - texture to use.
Intensity - how much distortion is applied.
Preview - blits distortion texture directly on camera without the need to submerge camera.



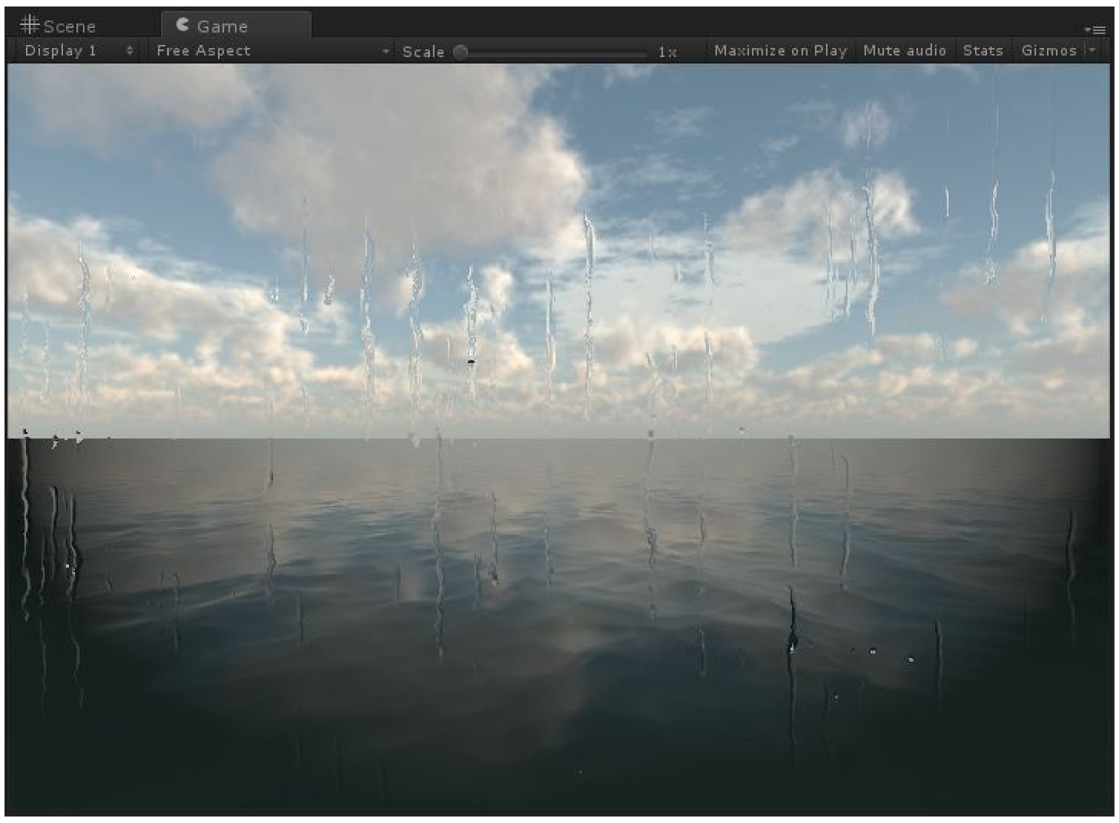
WATER RAINDROPS IME

Displays dynamic raindrops/water splashes on camera lens, that react to camera movement. *Water Raindrops* only renders the raindrops. You must create the raindrops either by accessing this component via script or using provided *Water Rain* component.

demo: demo scene presenting this feature can be found at *Ultimate Water System/Samples/Raindrops*

Parameters:

Force - external force acting on raindrops.
Volume Loss - how much mass the raindrops lose when traveling across the screen.
Resolution - the resolution, at which the raindrops are rendered.



Friction:

Air Friction - the force opposing raindrops movement.
Window Friction - the texture defining changes is window surface, causing the raindrops to change speed and direction across different sections of the screen.
Window Friction Multiplier - how much the window friction texture modifies the travel of raindrops.

Distortion:

Multiplier - how much the raindrops bend the light passing through them.
Normal Spread - how far apart are the points used for calculating normal vectors; higher values result in smoother surface.

Twirl:

Texture - this texture is used to twirl the screen uv's, causing more variation in raindrops movement.
Multiplier - how much twirl is applied.

Fade

Intensity - value of 1.0 causes raindrop tails to stay on screen forever. The smaller the value, the faster the tails fade.
Texture - additional texture that allows specifying different fade values on different parts of the screen.
Multiplier - how much the texture fade modifies base value.

Tracking:

Translation - how much force is inflicted on raindrops due to translational camera movement.
Rotation - how much force is inflicted on raindrops due to camera rotation.

WATER RAIN

Basic component that allows creation of the rain areas.

important: the *Cameras* game objects interacting with the *Water Rain* must have trigger colliders to work properly
note: this is a very basic script, but it shows the concept of how to use *Water Raindrops IME*, it can be used as a reference for your own emitters.

Parameters:

Intensity - how many raindrops are generated.
Size - the size range of the raindrops.
Life - how long the raindrops will stay on screen (if they stop along the way due to forces).
Force - the initial force of the raindrops when falling on camera lens.

WATER RIPPLES PROFILE

Defines the properties of the dynamically generated waves.
It can be create via *Create* menu in *Project View* by selecting *Create/Ultimate Water/Ripples Profile.*

Parameters:

Damping - how much the wave amplitude decreases with time.
Propagation - how fast the waves travel.
Gain - force inflicted by interacting objects.
Height Gain - wave amplitude decrease with depth.
Height Offset - wave amplitude decrease static offset.
Sigma - used for smoothing the calculated wave height map.
Noise Gain - how much noise is applied to damping.
Multiplier - multiplier for normal vector calculations.
Spread - smooths the calculated normals.

note: to use it in simulation, you must select it in either *Water Simulation Area* or *Water Simulation Space*.

WATER SIMULATION AREA

Defines the area, where the dynamic waves will be created from object interaction.
It can be created by right-clicking in *Hierarchy View* and selecting *Water/Simulation Area*, or added to existing object as a component.

The best way for creating simulation area (for a first-person game) is to attach the component under the Camera GameObject, slightly ahead of the player.

As the computations of the waves are computationally expensive, the created area should have textures the smallest size possible with acceptable results. The resolution is defined by width and height of the area times the *Pixels Per Unit* setting.

note: *Info* helpbox shows the current resolution of the textures that will be created.
note: *Dynamic* area loses information about created waves the moment the waves go outside its boundary.

important: the max resolution of the textures should be in the 1024-2048 pixel range.

Parameters:

Water - reference to the *Water Component*, which will be modified.
Profile - created *Water Ripples Profile*.
Type - is this simulation area will be moved at runtime:

- Static - area cannot be moved.
- Dynamic - [recommended] area can be moved, and all generated waves will be handled correctly.

PixelsPerUnit - [performance-critical] how dense is the simulation grid; higher density provides more detail of the waves, but is noticeably slower.
Depth Scale - [performance-important] used for detecting dynamic objects interaction with water (fish, skipping stones, etc.).
Enable Static Calculations -enables waves to be blocked by other objects
Static Depth Scale - [performance-important] used for reflecting waves from static objects (terrain, rocks, pillars, etc.). Ignored when *Enable Static Calculations* is disabled.

Edge fade:

Fade - enables waves amplitude damping close to the borders of the simulation area, to hide the transition between simulated and not-simulated water areas.
Fade Power - how much fading is applied.

WATER INTERACTIVE

Uses mesh renderers to create dynamic waves on the water surface.

demo: demo scene presenting this feature can be found at *Ultimate Water System/Samples/Dynamic Waves*

Parameters:

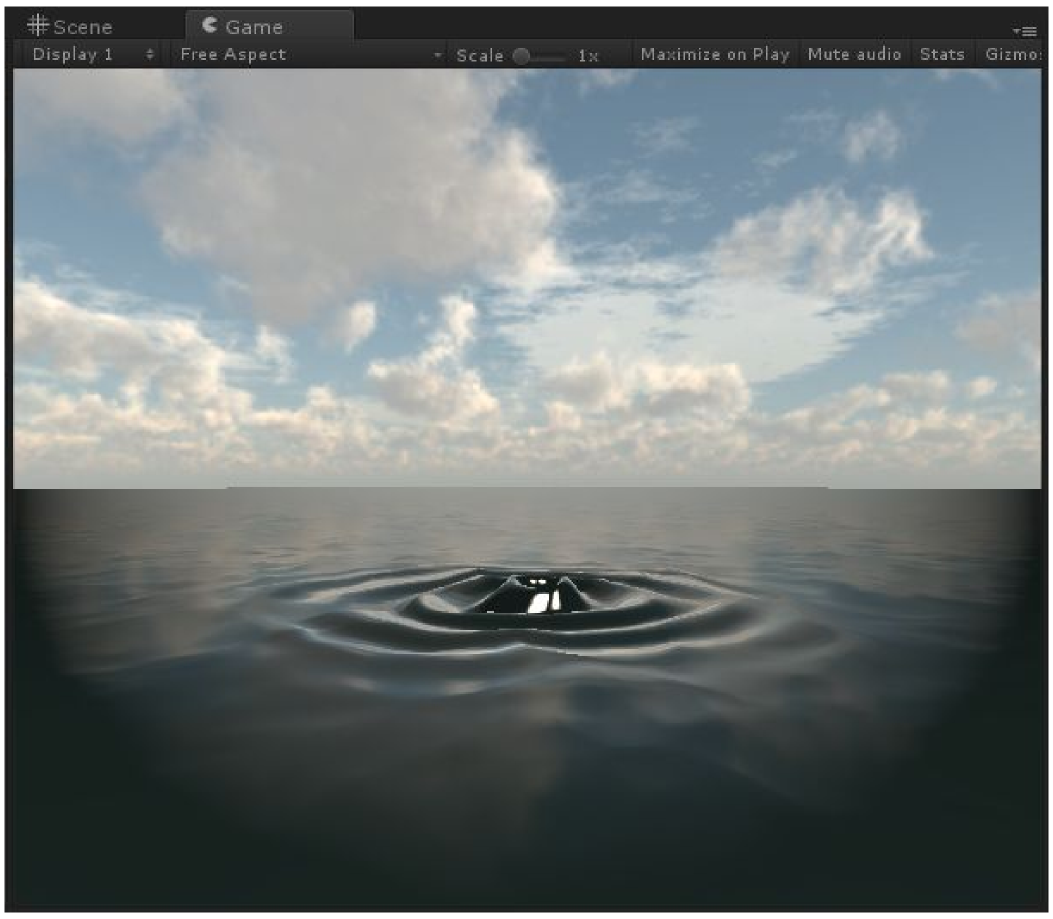
Multiplier - how much speed influences wave generation [0 - no interaction].

WATER FORCE

Applies force to the *Water Simulation Area* without the need of the mesh renderer.

Parameters:

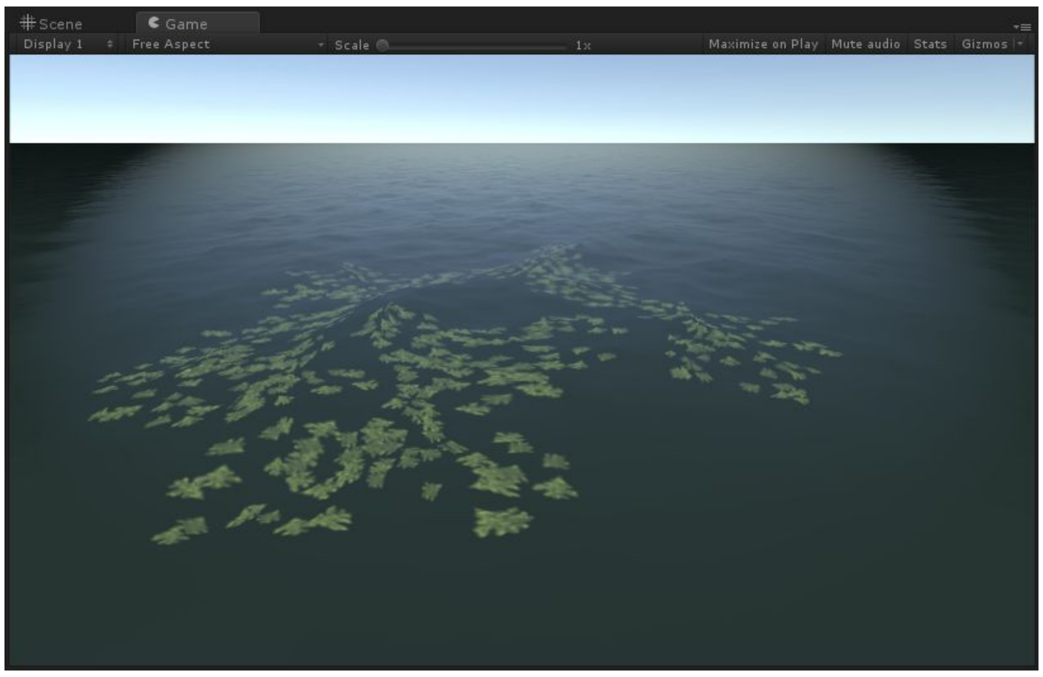
Force - the force added to water surface.



WATER PROJECTOR

Projects GameObject’s mesh renderer onto the water surface. It can be used to add water decals (dirt, leaves, etc.).
You can create it by right-clicking in hierarchy and selecting *Water/Projector*.

demo: demo scene presenting this feature can be found at *Ultimate Water System/Samples/Water Projector*



Parameters:

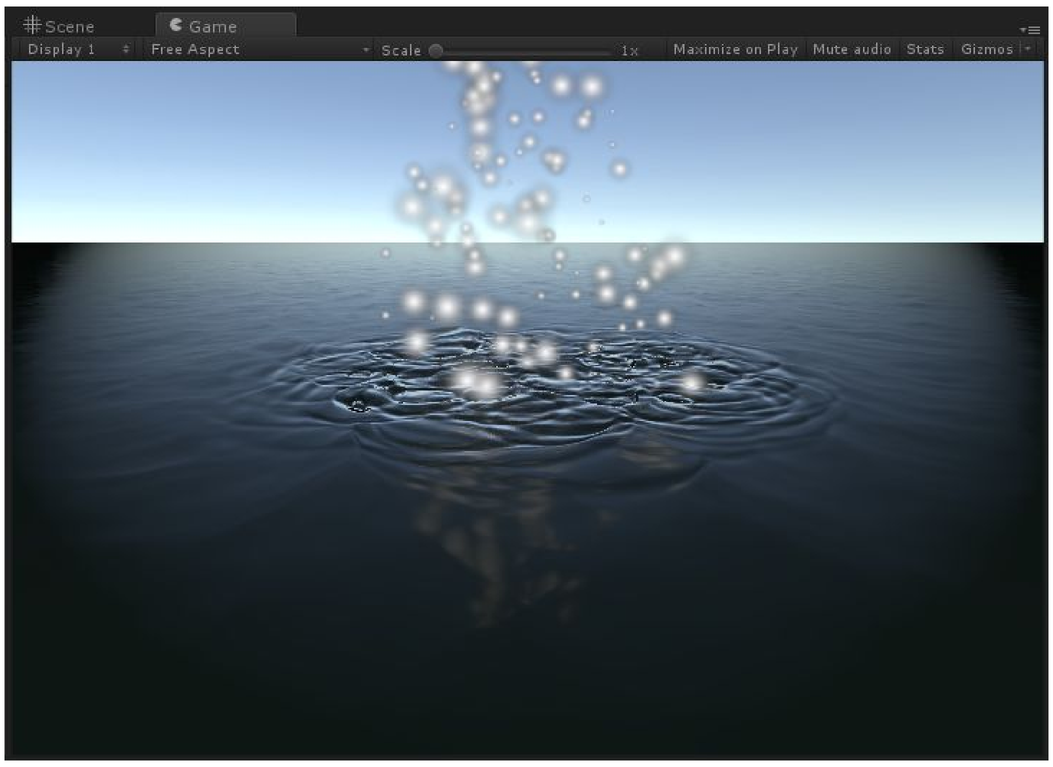
Type - defines the type of projection
- Diffuse: projects color texture
- Displacement: displaces water based on the grayscale height

Use Children Renderers - projects all meshes from that GameObject downwards the hierarchy.

Setup:

To create new *Water Projector:*
- Create new plane gameobject (select *Create/3D Objects/Plane* in the *Hierarchy* context menu
- In the object *Inspector* click *Add Component* and add *Water Projector* script
- Select either *Diffuse* or *Displacement* type:
    - Diffuse - *Material* attached to the *Mesh Renderer* will be used
    - Displacement - You must reference the *Material* that uses *Water/Overlay/Displacement* shader

WATER PARTICLE DISPLACEMENT



Detects when particles collide with water and adds force to create waves.

demo: demo scene presenting this feature can be found at *Ultimate Water System/Samples/Particles*

Parameters:

Water - reference to Water script.
Force - how much the particle disturbs water surface.
Used Particles - percentage count of the particles causing waves.
Force Over Speed - how much particle speed changes the force inflicted on water surface.

Size Multiplier - how much particle size changes the force inflicted on water surface.
Frame Split - [advanced] splits the calculation
of particle-water collisions between frames,
lowering the cpu-usage at the cost of precision.

SHADER SET

Composes shaders to match selected water generation options.
Only the necessary features should be enabled, this allows for smaller shader generation that results in faster calculations.

Reflection & Refraction:

Transparency Mode - does the light should pass through water:

- Solid - water is opaque
- Transparent - water is transparent

Reflection Probe Usage - defines the reflection probe type usage type:

- Off - neither reflection probes nor skybox will be used,
- Blend Probes - only the blend probes will be used,
- Blend Probes And Skybox - both types supported,

Planar Reflections - determines the type of planar reflections

- Disabled - no planar reflections will be used, overriding *Water* settings
- Normal - faster, but inaccurate,
- High Quality - slower, but more precise.

Receive Shadows - should the water receive shadows from other objects

Waves:

Wind Waves Mode - global override of the *Water* waves mode setting.
Dynamic Smoothness Mode
Local Effects Supported - when turned off, disables local effects like water projectors, displacement.
Foam - toggles foam rendering.

Render Modes:

Defines, what render modes will be supported by water.

Geometry Support:

Projection Grid
Custom Triangular Geometry

Volumes:

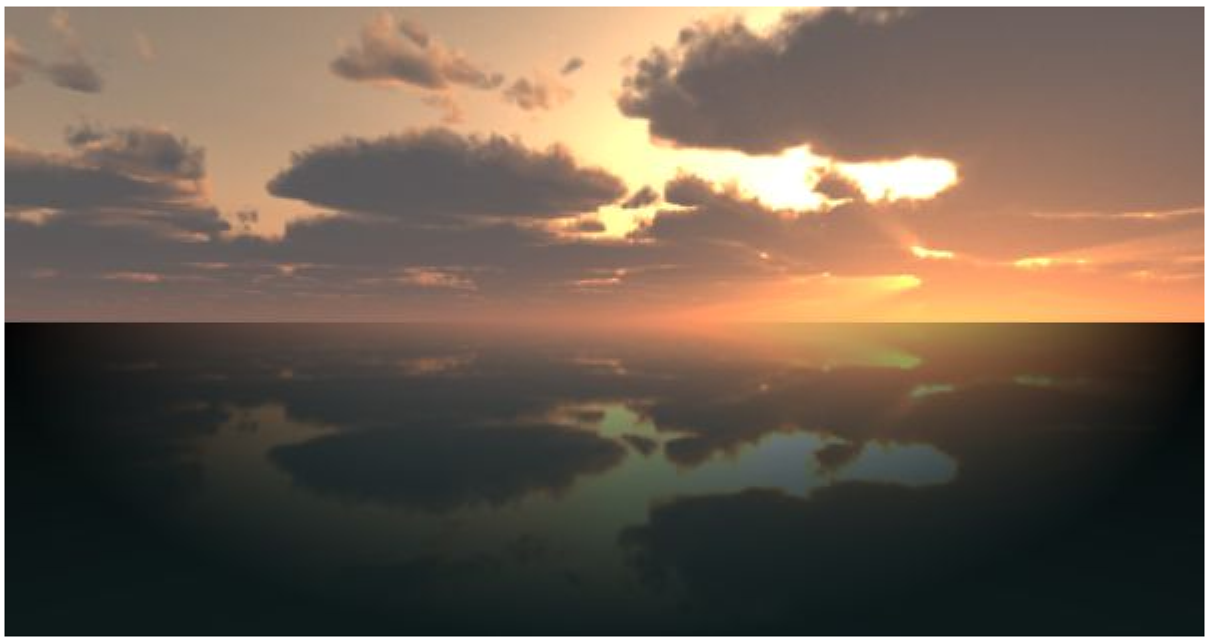
Display Only In Additive Volume
Waves Align

Surface:

Normal Mapping Mode
Support Emission

COMMON ISSUES

UNWANTED STRONG VIGNETTE



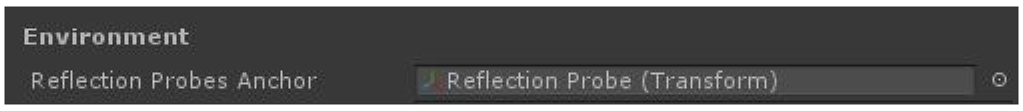
After setting up the water, you may encounter strong vignette-like effect.
The cause of the problem lies in the fact, that the *Shader-Set* selected has *Reflection Probe Usage* turned on, but no Reflection Probes are found on scene.

If you do not want to use Reflection Probes:

- Select *Water Component* and click *Shader-Set Edit* button.
- Locate *Reflection Probe Usage* and select *Off* from the dropdown.
- Click the *Apply* button which appeared after changing the *Shader-Set* options.

If you do want to use Reflection Probes:

- If you are using either *Default Queue* or *Image Effect Forward* in your *WaterCamera* - create just one reflection probe that spans over the all the water surface. Be sure to link the transform of the Reflection Probe to the *WaterComponent*



Environment

Reflection Probes Anchor: Reflection Probe (Transform)

- If you are using *Image Effect Deferred*, then all the probes in the scene will be blended correctly, so ensure that you have Reflection Probes covering the desired water surface

MANY-WATER SCENES ARE SLOW

There are some important settings that can slow down the scene if not handled correctly:

- Ensure that *Water Components* share the *Spectrum Data* through *CopyFrom* parameter.
- When using *Volumes Add*, use a *custom mesh* instead of *radial geometry*. This prevents from rendering full-screen water pass just to mask it immediately after.

MYSTERIOUS COLLIDERS

Ultimate Water System internally uses trigger colliders near camera and objects with enabled water physics to detect when they get close to a water body without hurting performance. To ensure that this won't affect your game:

- Ensure that their layer is excluded from Lens Flares occlusion (it is by default).
- Exclude this layer from all your scripting raycasts.

By default, all water colliders are placed on Layer 1 (*TransparentFX*). You can change this in *Edit/Project Settings/Water.*
If you use UFPS from the Asset Store, please open vp_Layer.cs and add this layer to ignore list. There may be some other assets that may have issues with this. It is always easily fixable – please contact us to get a support.

UNSTABLE SHIPS

Cause:

PhysX computes center of mass with assumption that body density is constant. This is correct most of the time, but not a case for ships. In real world, ships are balanced to have their center of mass as low as possible to gain stability.

Solution:

- Click “Materialize Center of Mass” context menu item on ship’s rigid body.
- Find center of mass object among ship children in scene hierarchy.
- Move it down.

UNDERWATER IME WITH CONTAINED WATER

When using *Underwater IME* with water contained in *Additive Volumes* and/or using *Custom Meshes* the underwater effect is not enabled when it should be (i.e. the camera is under the water, but the effect is disabled).

Cause:

This is caused by the fact that the calculations determining when the *Camera* is submerged in water are dependent of the type of the *Water*. The calculations for *Boundless* water are simpler, but do not account for the *Additive Volumes*.

Solution:

Make sure that the *Boundless* toggle in the *Water Component* is unchecked.

COMPATIBILITY ISSUES WITH IMAGE EFFECTS

Ultimate Water System should be compatible with most, possibly all, image effects, but opaque image effects may cause some issues on some setups that should be easily fixable.

Solution:

Change your water camera render mode from *Default Queue* to some other mode. *Default Queue* is the fastest render mode, but not compatible with some external effects. *Image Effect Forward* is pretty much equal performance-wise, but is compatible with most opaque image effects, while *Image Effect Deferred* is a bit slower in simple scenes, but should be compatible with all image effects, even those that access GBuffer.