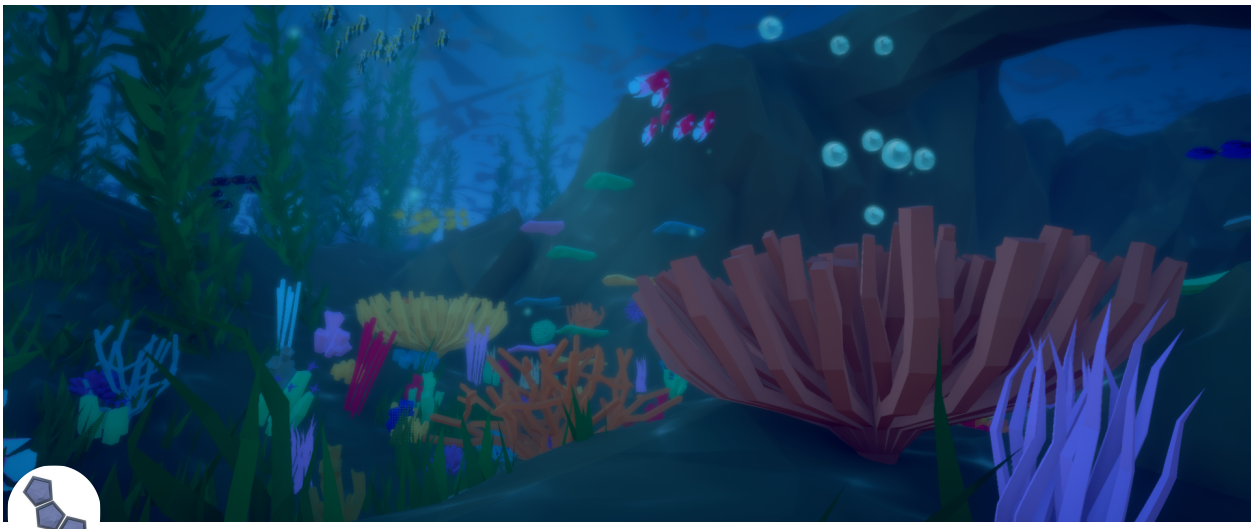


Low Poly Underwater Pack Documentation

Information

- **Asset Version -** v1.0.2
- **Author -** Ventuar3D
- ▼ **Contact Information:**
  - **Email** - ventuargames@gmail.com
  - **Twitter** - @Ventuargames
- Patch Notes Link

Table of Contents

Introduction

Hi there, and thank you for purchasing the **Low Poly Underwater Pack**! And if you are here for some other reason, I appreciate you taking a look at my asset nonetheless 😄

In this page you will find details on how to set up and use the various tools and features this pack offers. I am proud of how this pack came out and the work that has been put into it, and I hope you will feel satisfied by how much this pack offers to suit your low poly underwater needs!

If there is anything in this page that is unclear, or if there are any problems, bugs, or new features you'd like to see in the pack, feel free to contact me via either my email or Twitter using the links at the top and bottom of this page. I will try to respond to and fix issues as soon as I can, but I am also a full-time student so my time to address them may be limited.

If you feel comfortable enough to try and fix any problems that may arise as you are using the pack yourself, I have tried to include as many helpful inline comments as I could to help you effectively understand and debug the code. Seeing as this page is in Notion (unless you are viewing the PDF version), you should also be able to comment directly in here if there are any errors or issues with clarity in this documentation.

If you enjoy the pack, it would help me out a lot if you left a positive review in the Unity Asset Store or CGTrader so that the pack more easily can reach others! Other than purchasing the pack, reviews are the best

way you can support me and the content I create. Thank you! 🐟🌊

Feature Overview

- 148 Models
- Water
- Underwater effects
- Procedural animation (fish, corals, and plants)
- Customizable ships
- Fish AI + Boids
- Buoyancy
- Modular Terrain
- Game-ready demo scene
- Full Built-in RP + URP Compatibility

> 💡 Note that all screenshots and gifs shown in this page were taken in the 2020.3.13f1 editor version. User interface and property locations are subject to change based on the version in-use. Also note that GIFs will not work when viewing the PDF version of this documentation.

Importation

Important Project Settings

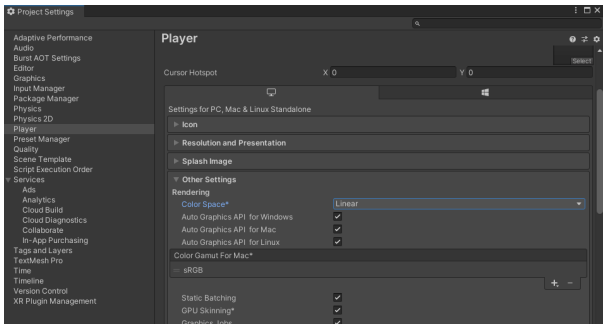
There are a number project-level parameters which can be changed to achieve desired visual results:

Color Space

- **Path:** *Edit > Project Settings > Player > Other Settings > Rendering > Color Space*

The water shader currently requires a linear color space to look correctly. Water in the gamma color space should appear completely as the foam color, which is currently a bug with the shader.

Your project may already by in a linear color space, in which no change is required. If you are not sure or currently have your project in a gamma color space, however, go to the **path** and change "Color Space" from Gamma to **Linear**.



The Player subsection of Project Settings, in which the option to change Color Space is located.

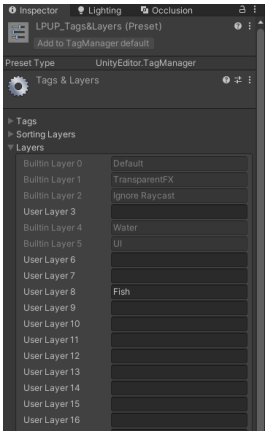
Layers

- **Path:** *Edit > Project Settings > Tags and Layers*

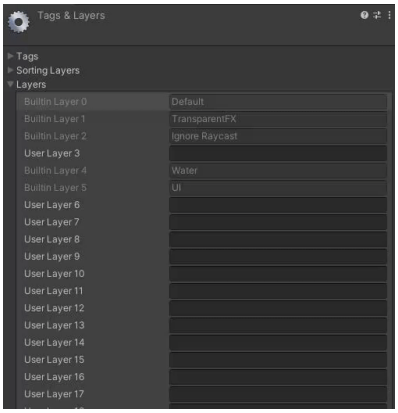
The fish AI require their layers to be set as Fish to work correctly. By default, this layer is set to User Layer 8.

If you do not already have any predefined tags, there is a preset *(see image)* located in the "UnderwaterPack" folder *(Ventuar > UnderwaterPack > LPUP_Tags&Layers)* which, if applied to the project, will result in the fish AI working correctly. To apply this preset to the project, to go the **path** and apply the preset using the preset icon in the top right corner.

If you are unable to apply the preset, go to the **path** and create a new layer named **"Fish"** *(case sensitive)*. The fish AI will automatically apply this layer to themselves on startup.



The default Tags & Layers preset for the Low Poly Underwater Pack, containing the "Fish" layer used by fish AI in User Layer 8.



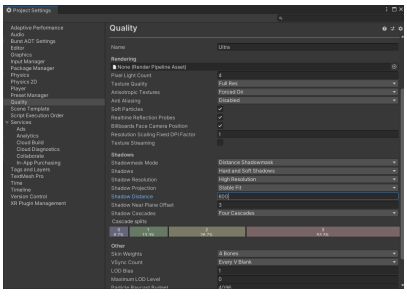
A demonstration of how to apply the LPUP_Tags&Layers preset to your project.

> Note that if you do not make the fish layer in User Layer 8, the fish prefabs will have incorrect layers. This should not affect gameplay as the fish will automatically apply the correct layer to their respective instances, but may result in some confusion if not corrected.

Shadow Distance

- **Built-in RP Path:** *Edit > Project Settings > Quality > Shadows > Shadow Distance*
- **URP Path:** *Universal Render Pipeline Asset > Shadows > Max Distance*

A change in shadow distance is not required, but may result in incorrect or unintended visuals in the Demo scene in some scenarios. The water object is a shadowcaster, and many of the visual issues result in how much of a shadow it is able to cast.



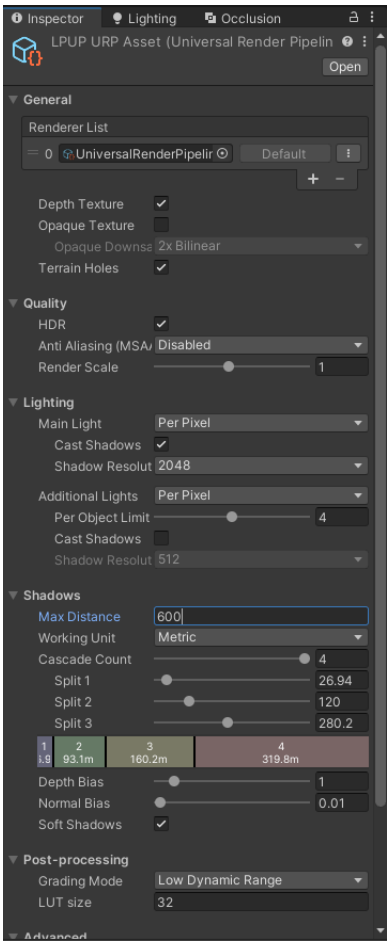
With a too low shadow distance, shadows will not project far enough to reach the underwater fog effect, resulting in the shadow draw cutoff being easily visible.

When moving above the surface of the water, a too low shadow distance may result in the shadow draw cutoff being visible below the water.

From experimentation, a shadow distance of **300** is enough for the underwater effect to appear correctly, and a shadow distance of **600-1000** is ideal for above-water rendering in the Demo scene.

To change the shadow distance, go to the **path** respective to your render pipeline and change the value to your desired value. The default value for URP is **600**.

The Quality subsection of Project Settings, in which the option to change shadow distance is located in the Built-in Render Pipeline.



The default Universal Render Pipeline Asset included in the URP_EXTRACTME package named "LPUP URP Asset", where the option to change the max shadow distance for URP is located.

LOD Bias

- **Path:** *Edit > Project Settings > Quality > LOD Bias*

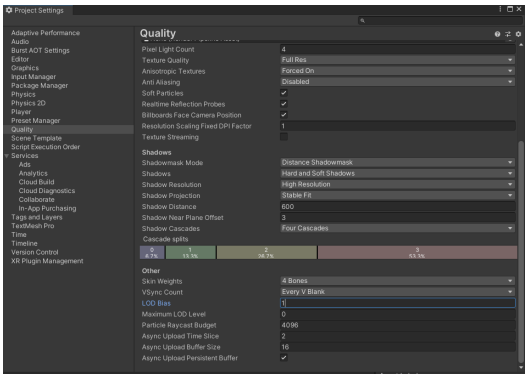
Most objects in the Low Poly Underwater Pack use LODs with a dithering fade-to-cull to optimize performance. A change in LOD Bias is not required, but may result in undesired visual results in the Demo Scene.

The default LOD Bias value for a project is often 2. The LODs in the Low Poly Underwater Pack have been configured to look correct with a LOD Bias value of 1 in

order for the fade-to-cull transition to be masked by the underwater fog effect.

Keeping the LOD Bias value at 2 or above will result in the transition to cull being visible before the objects reach the underwater fog. It will also result in a performance decrease in the Demo scene.

In order to change the LOD Bias value, go to the **path** and change the value to **1**.



The Quality subsection of Project Settings, in which the option to change the LOD Bias is located.

Built-in Render Pipeline

The Low Poly Underwater Pack should function properly without errors upon importation. However, there are several things you should do and be aware of when working in this pipeline:

Packages

There are two packages you should import if you want the pack to work smoothly and as intended in the Built-in Render Pipeline:

▼ **Post Processing**

- Used to create both surface and underwater post processing effects for visual and artistic purposes.

▼ **Burst**

- Used to significantly speed up boids calculations by utilizing the Burst Compiler and C# Jobs System.

To download packages, go to *Window > Package Manager*, and in the top left corner, navigate to *Packages: Unity Registry* (or *All packages)* and search for the above packages.

Note that the package will still function correctly without these packages, but will lack the features listed above. Some warnings may appear when working with the package without these packages downloaded, but they can be ignored.

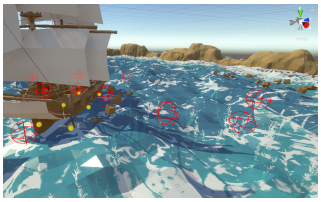
Forward vs Deferred Rendering Paths

The current rendering path can be changed in the **Camera component** of the main camera.

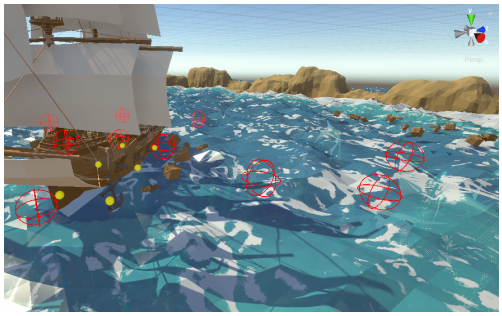
- Shadows on the surface of the water behave differently depending on the rendering path in use.
  In the forward

rendering path, shadows in cascade 0 will not appear on the water *(see picture)*. This is caused by a bug in forward rendering that affects the way the water renders shadows. Due to this, I recommend using the **deferred rendering path** when using the Built-in Render Pipeline.

  - If you wish to use the forward rendering path, one solution to this issue would be to remove cascade 0. However, this will result in lower-quality shadows up-close.



Water shadows in the forward rendering path—notice the circular gap of missing shadows in cascade 0.



Water shadows in the deferred rendering path.

- Due to errors in the way Unity handles getting light direction in a shader while the camera is using the deferred rendering path in the Build-in Render Pipeline, the **underwater caustics will not react to changes in the light direction when using the deferred rendering path.**
  - This issue is not present when using the foward rendering path, but due to the previous point, I reccomend using deferred.
  - If you wish to change the caustics projection angle manually, you may do so in the Causics.shader file. The area to do so is clearly marked *(see image)*.

```
139
140                 // Custom logic for if the camera is in deferred. _WorldSpaceLightPos0 is bugged/inconsistent in the built-in deferred rendering path, so we just make the direction fixed.
141                 /*
142                  *  CHANGE THE CAUSTICS PROJECTION ANGLE HERE
143                  */
144 #if UNITY_PASS_DEFERRED
145                     float3 projAngle = float3(0,-.5f,.2f);
146                     causticProj = max(0, dot(reflect(projAngle, normal), normal));
147 #endif
148
```

Code snippet of Caustics.shader where the caustics projection angle can be manually changed.

Universal Render Pipeline

When using the Universal Render Pipeline, there are a few things you need to do and be aware of before you can use the Low Poly Underwater Pack:

Packages

There are two packages you should import if you want the pack to work smoothly and as intended in the Universal Render Pipeline:

▼ **Universal RP**

- Required to use the Universal Render Pipeline.

▼ **Burst**

- Used to significantly speed up boids calculations by utilizing the Burst Compiler and C# Jobs System. The package will still function correctly without Burst, but will lack the features listed.

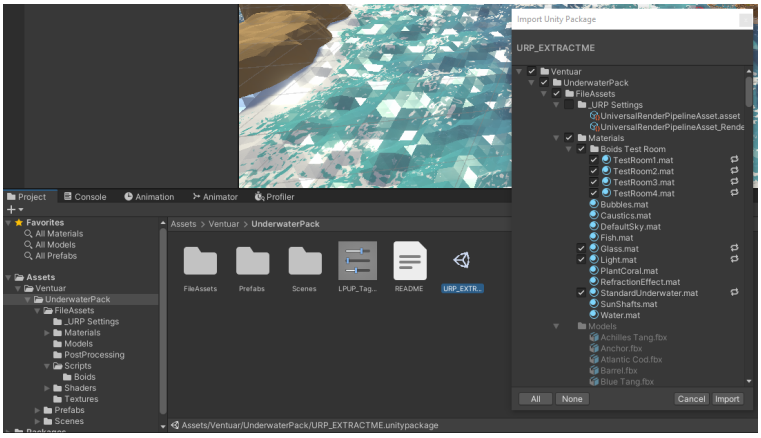
To download packages, go to *Window > Package Manager*, and in the top left corner, navigate to *Packages: Unity Registry* and search for the above packages.

URP_EXTRACTME

Some of the assets in the Low Poly Underwater Pack do not transfer seamlessly between render pipelines.

To gain access the assets needed for the pack to work in the Universal Render Pipeline, there is a unitypackage file included in the "UnderwaterPack" folder *(Ventuar > UnderwaterPack > URP_EXTRACTME)*. If you are working within the Universal Render Pipeline, **you must import this package before the pack can be used** *(see picture)*.

Once the package has been imported, all of the assets within the Low Poly Underwater Pack should work correctly within the Universal Render Pipeline.



The location and contents of the URP_EXTRACTME package.

Quality & Graphics Settings

- **Quality Path:** *Edit > Project Settings > Quality > Rendering*
- **Graphics Path:** *Edit > Project Settings > Graphics > Scriptable Render Pipeline Settings*

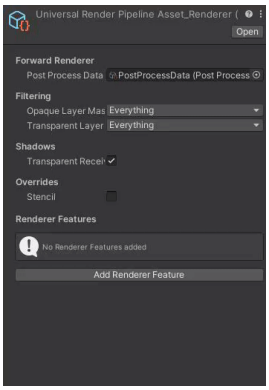
Ensure you go to the two **paths** and assign your Universal Render Pipeline Asset to its respective properties. As mentioned in the section below, there is a Universal Render Pipeline Asset included with the pack *(Ventuar > UnderwaterPack > FileAssets > _URP Settings > LPUP URP Asset)* that includes all the settings needed for it to work optimally.

Universal Render Pipeline Asset Settings

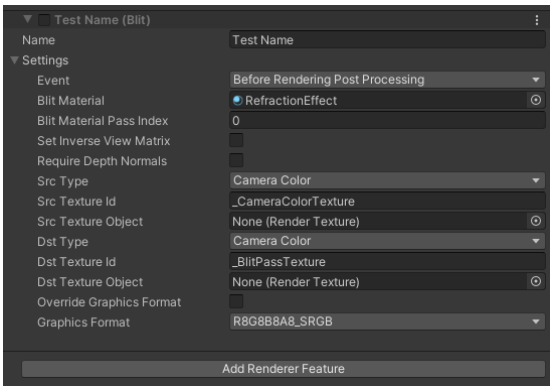
The Low Poly Underwater Pack contains a default Universal Render Pipeline Asset named "LPUP URP Asset" *(Ventuar > UnderwaterPack > FileAssets > _URP Settings > LPUP URP Asset).* However, using your own Universal Render Pipeline Asset is perfectly fine.

Most of the visual features of the pack should work correctly with a stock Universal Render Pipeline Asset. However, in order to utilize the underwater refraction/distortion feature, some steps must be taken for it to work properly *(see images for an example)*:

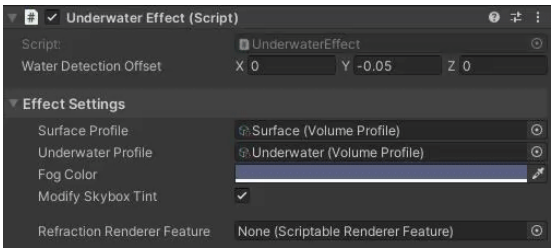
1. In your Universal Render Pipeline Renderer asset *(by default named yourassetname_Renderer)*, click on **Add Renderer Feature**, and select **Blit**.
2. **Rename** this renderer feature to your desired name. In the default Low Poly Underwater Pack URP renderer, it is called "Refraction Feature", but in this example it is called "Test Name".
3. Change the "Event" property to **Before Rendering Post Processing**.
4. Assign the **RefractionEffect** material to the "Blit Material" Property.
5. In the Demo scene, select the "Main Camera" object, and in the "Underwater Effect" component, **assign your new renderer feature** to the property named "Refraction Renderer Feature".



Step 1.



Steps 2, 3, and 4.



Step 5.

You may also want to change shadow settings to better suit the Demo scene. Refer to the above section on shadow distance settings to see the reccomended shadow distance values for the Demo scene. You may use as many or a few shadow cascades as you'd like, but I reccomend using **all 4 cascades** for the Demo scene.

Note that if visual elements of the pack don't look correct in URP with your own Universal Rener Pipeline Asset, try enabling the **HDR** and/or **Depth Texture** properties of your asset if not already enabled.

High Definition Render Pipeline

The Low Poly Underwater Pack currently does not support HDRP. However, with some tweaking, the models, textures, and boids AI will still be usable; the shaders, post-processing, buoyancy system, water, and procedural animations will not.

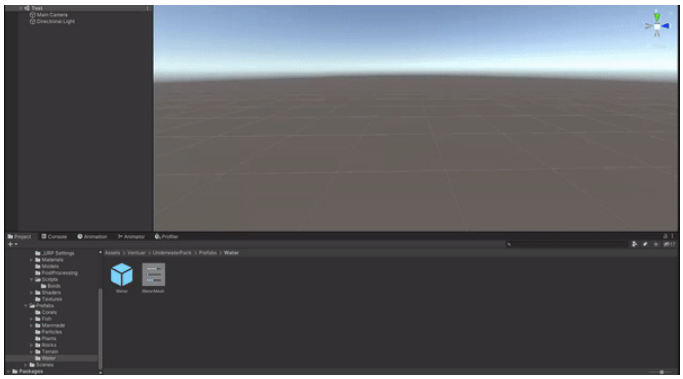
Usage

Here you will find details on how to use each major feature of the Low Poly Underwater Pack. If you ever get confused, each serialized script property in the Low Poly Underwater Pack contains a tooltip to help you effectively utilize it.

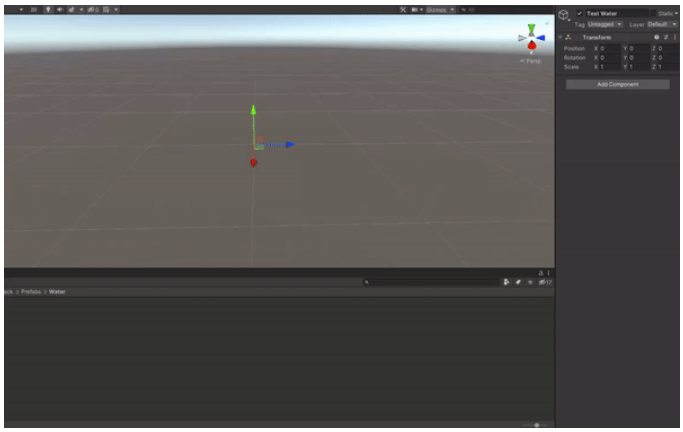
Water

There are two ways to create a new water object:

1. Drag and drop the **Water** prefab *(Ventuar > UnderwaterPack > Prefabs > Water > Water)* from the project window into your scene.
2. Add the **WaterMesh** script onto any game object. The script will automatically add any necessary components for the water to work correctly.



Method 1 of creating a water object, dragging and dropping the water prefab from the project window into the heirarchy.



Method 2 of creating a water obejct, adding the WaterMesh script to an empty game object.

The WaterMesh script is the largest script in the Low Poly Underwater Pack. Each property contains a tooltip for you to use if you are unsure what it does, and they are grouped into headings for your conveinence:

▼ **Visualization Settings**

- Contains options to display a wireframe visualization of the water plane, as well as options to turn on and off various water features including waves, noise, and foam.

▼ **Mesh Settings**

- Contains the x-z size and subdivision properties of the water plane. Toggling "Is Equilateral" will you allow you to modify the size of the water plane as a square.

▼ **Shader Setting**

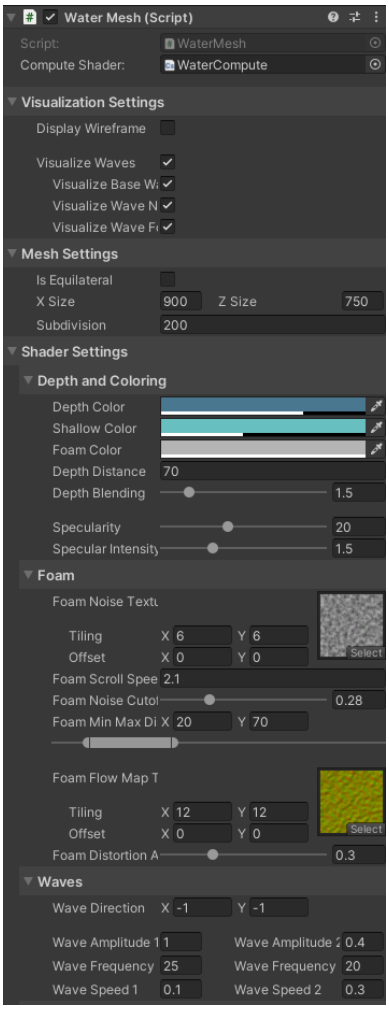
- Contains properties and sub-headings that affect the visual representation of the water.

  ▼ **Depth and Coloring**

  - Contains properties needed to control the coloring and lighting properties of the water at various depths, as well as values to fine-tune how deep you are able to see into the water.

  ▼ **Foam**

  - Contains all properties needed to control the water foam besides its color, which is contained in the above "Depth and Coloring" section. Includes places to



Serialized properties in the "WaterMesh" component.

provide your own foam and flow map textures.

▼ **Waves**

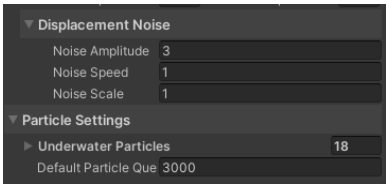
- Contains all properties needed to control both the primary and secondary waves.

▼ **Displacement Noise**

- Contains properties needed to control the noise overlayed on top of the waves.

▼ **Particle Settings**

- Contains properties needed to help particles render correctly underwater.



Serialized properties in the "WaterMesh" component, continued.

💡 Note that some particles may render in at the incorrect depth when looking through the water *(i.e. the particle is underwater, you are above water, and the particle may render in front of the water. Vice versa can also occur)*. To fix this, WaterMesh contains logic to adjust the render queue of underwater particles provided they are all located under a single parent object. Assign this parent object containing underwater particles to the **Underwater Particles Parent** property to rectify this potential issue. A better fix for this issue may come in a future update.

Fish

Single Fish

Single fish are any fish prefabs with the **SingleFishAI script** attached. Single fish prefabs *(Ventuar > UnderwaterPack > Prefabs > Fish > Single)* can be dragged and dropped into any scene. They will interact with any and all colliders that are not tagged as "Fish", meaning that they can "swim" even if no water is present.

Each property in the SingleFishAI script contains a tooltip for you to use if you are unsure what it does, and they are grouped into headings for your conveinence:

▼ **General Settings**

- Contains general properties, options, and data visualization options of the fish, including whether the fish swims straight or in a bobbing motion.

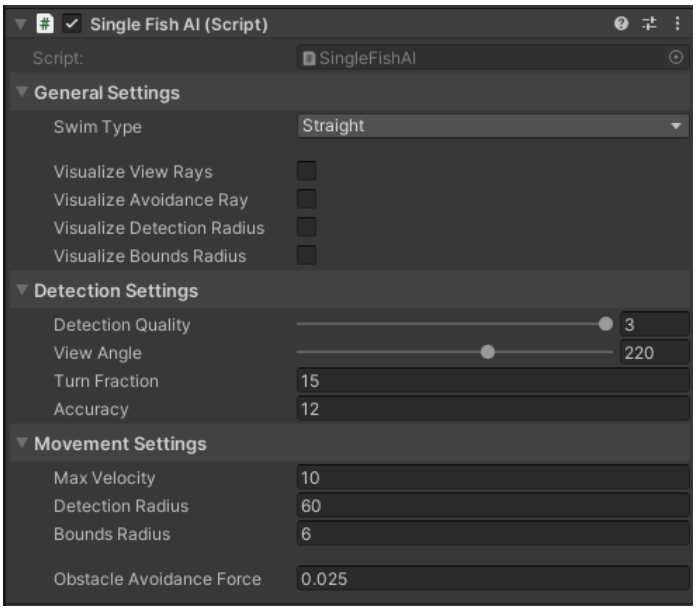
▼ **Detection Settings**

- Contains properties which control the accuracy and

degree to which the fish can detect its surroundings and avoid obstacles.

▼ **Movement Settings**

- Contains properties which control the ways with which the fish moves and reacts to its surroundings.



All serialized properties in the "Single Fish AI" component.

Note that obstacle detection works in a way which if every direction in a fish's view is blocked by an obstacle inside the detection radius, it will travel forward as a failsafe, often resulting in the fish traveling through obstacles or out of bounds. Single fish prefabs have larger detection radii and lower obstacle detection forces, meaning that the mentioned effects are likely to occur in small enough spaces. If you have a smaller scene which you want to place single fish into, you could decrease the detection radius, increase the obstacle avoidance force, and/or decrease the max velocity of the fish to prevent this.

Boids

Boids, or flocking fish, are any fish prefabs with the **FlockingBoidAI** script attached. Much like the single fish, flocking fish prefabs *(Ventuar > UnderwaterPack > Prefabs > Fish > Flocking)* can be dragged and dropped into any scene. They will interact with any and all colliders that are not tagged as "Fish", meaning that they can "swim" even if no water is present.

They will flock together with fish of the same type by using a boids algorithm. The algorithm they use will depend on if the **Burst** package is installed and the "Run With Jobs" option is checked in Boid Master. If it is, the flocking fish will utilize an optimized algorithm utilizing the C# Jobs System and Burst Compiler. If not, they will use a slower version of that algorithm.

Each property in the FlockingBoidAI script contains a tooltip for you to use if you are unsure what it does, and they are grouped into headings for your conveinence:

▼ **General Settings**

- Contains general properties, options, and data visualization options of the fish,

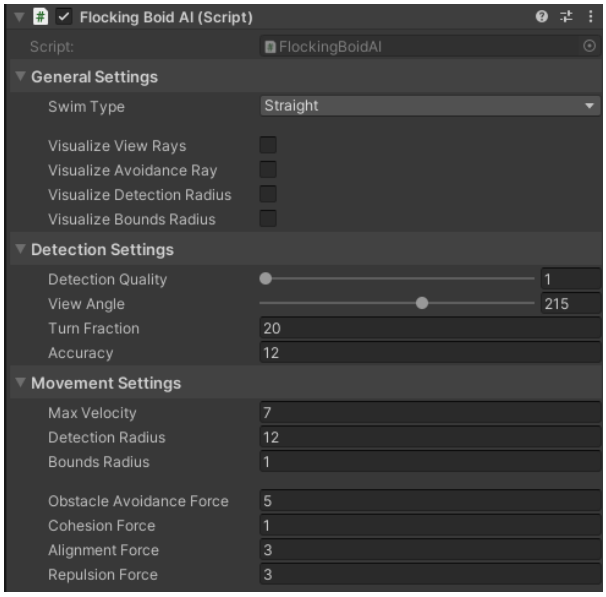
including whether the fish swims straight or in a bobbing motion.

▼ **Detection Settings**

- Contains properties which control the accuracy and degree to which the fish can detect its surroundings and avoid obstacles.

▼ **Movement Settings**

- Contains properties which control the ways with which the fish moves and reacts to its surroundings, including how it reacts and flocks with other fish.



All serialized properties in the "Flocking Boid AI" component.

Boid Master

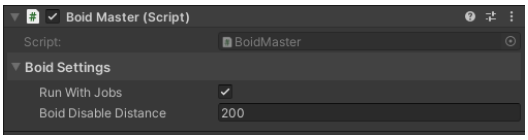
The **BoidMaster** script is an optional utility for boids, or flocking fish, that offer a couple different settings to tweak how they function:

▼ **Run With Jobs**

- A toggle that enables the use of the more efficient C# Jobs boids algorithm. This setting will only appear when the Burst package is installed, as the C# Jobs algorithm requires the Burst compiler to function. This is highly reccomended in order to improve the performance of the flocking fish.

▼ **Boid Disable Distance**

- The distance at which boids will be culled and disabled in the scene. Lowering this value can help improve performance at the cost of some naturality.



All serialized properties in the "Boid Master" component.

Boid Spawners

Boid Spawners are objects that can contain the **BoidSpawner** script, which allows the obejct to spawn fish in a radius around itself on startup. This can be used to effectively and naturally populate a scene.

▼ **Has Bounding Box**

- Toggle to indicate whether or not the spawner will have a bounding box. If true, the spawner will

create a box collider in which the fish spawned by that spawner will stay inside.

- Upon toggle, the spawner's layer will be changed to "Fish", meaning that the fish spawned by the spawner can detect it via special logic, but other fish will not be affected by it.

▼ **Fish Prefabs**

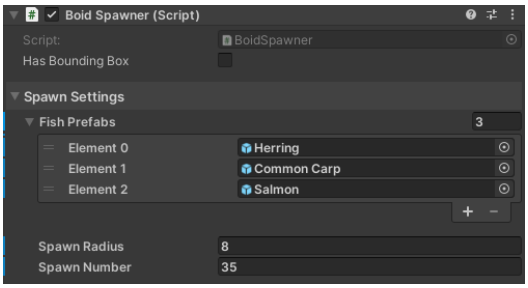
- The number of fish prefabs to be spawned by this spawner. Each fish will be spawned a randomly within the spawn number limit *(i.e. a spawn distribution given the settings in the picture could be 12 herrings, 15 common carps, and 8 salmon)*.
- Any number of fish prefabs can be selected to spawn.

▼ **Spawn Radius**

- The radius around the object within which the spawned boids will be randomly placed on startup.

▼ **Spawn Number**

- The number of boids the spawner will spawn.



All serialized properties in the "Boid Spawner" component.

Note that boid spawners can also spawn single fish. However, the spawners will not spawn any other object other than ones with either the Single Fish AI or Flocking Boid AI components.

Animation

All fish are procedurally animated using a script called **FishAnimation**. This script is placed on the child object of the fish where the mesh components are also located.

FishAnimation works by adding 2 different sin wave distortions to the mesh's vertices—one for the front and back of the fish—with customizable distortion axes for each. The script contains properties for controlling these waves and axes, with each grouped into headings for your convenience:

▼ **General Settings**

- Contains options for determining the main and secondary distortion axes as well as wave visualizations. Wave visualizations occur by changing the color of the mesh to a grayscale representation,

with a black color signifying no distortion and a white color signifying full distortion.

▼ **Wave 1 Settings**

- Contains properties which control the various aspects of the main distortion wave.

▼ **Wave 2 Settings**

- Contains properties which control the various aspects of the secondary distortion wave.

▼ **Misc. Settings**

- Contains miscellaneous properties for procedural fish animation, including the wave gradient clamping and how randomized each instance of this object can be.

All serialized properties in the "Fish Animation" component.

Buoyancy

Objects have the ability to float in the water using the **Buoyancy** script. These buoyant objects can be dragged and dropped into any scene, and if an object with a WaterMesh script is in the scene, it will react to it as a buoyant physics object. The Buoyancy has a number of properties to help control how it behaves in the water:

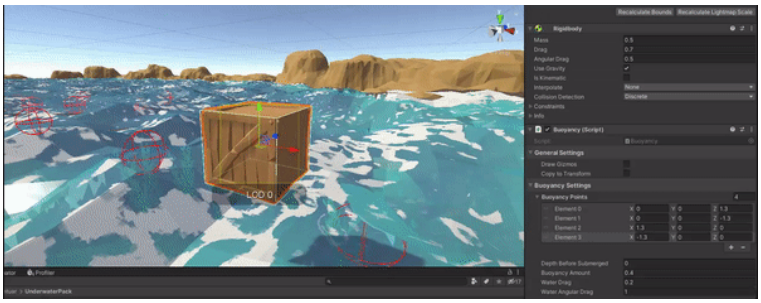
▼ **General Settings**

▼ **Draw Gizmos**

- Enable to draw a visualization of the buoyancy points on the object, with the gizmo size being controlled by the value in the **Gizmo Size** field *(see top gif)*.

▼ **Copy to Transform**

- Enabling this will allow you to copy the transform of this object to another transform assigned in the



Tweaking buoyancy values to see how the buoyancy point visualization changes.

Copy Transform field after physics calculations.

▼ **Buoyancy Settings**

▼ **Buoyancy Points**

- The local-space points with which the object will use to calculate buoyancy. These points will have buoyancy forces applied to them locally when underwater.

▼ **Depth Before Submerged**

- The depth each buoyancy point can reach before it is considered submerged

▼ **Buoyancy Amount**

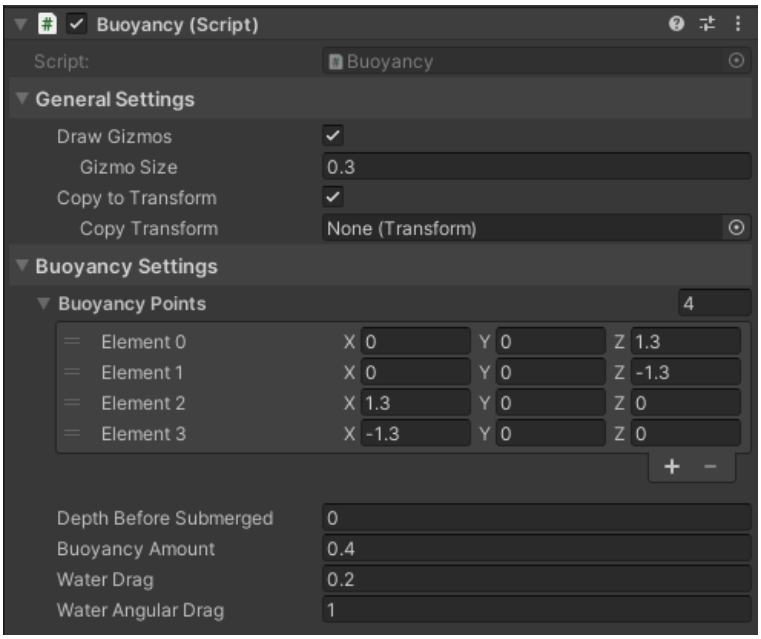
- How buoyant the object is, or how forceful the buoyant forces will be applied to each buoyancy point.

▼ **Water Drag**

- The amount of drag the object experiences while submerged.

▼ **Water Angular Drag**

- The amount of angular drag the object experiences while submerged.



All serialized properties in the "Buoyancy" component.

Note that buoyancy will work even if there are no water objects in the scene, and will also be able to tell which water object to use if there are multiple in the scene.

BuoyancyMaster

The **BuoyancyMaster** script is an optional utility for buoyant objects that offer a couple different settings to tweak how they function:

▼ **Use Accurate Detection**

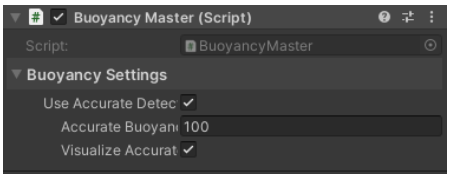
- Enables use of a compute shader algorithm to compute accurate water detection for buoyancy calculations. Can often by heavy on performance if overused. When buoyant objects are not using accurate detection, they use a faster approximate algorithm that does not always yield accurate results up-close.

▼ **Accurate Buoyancy Dist**

- The distance at which buoyant objects will switch from using approximate to accurate water detection. This property will only appear if **Use Accurate Detection** is toggled.

▼ **Visualize Accurate Detection Objs**

- Enables the runtime visualization of which objects are currently and actively using accurate detection.



All serialized properties in the "Buoyancy Master" component.

Plants and Corals

Plant and coral prefabs can be placed in a scene just like any other prop. However, they contain the **PlantCoralAnimation** script which procedurally animates them in the scene

PlantCoralAnimation works by adding a sin wave distortion to the x and z values of the mesh's vertices. The script contains properties for controlling this wave as well as some custom properties for coral colorization, with each grouped into headings for your convenience:

▼ **General Settings**

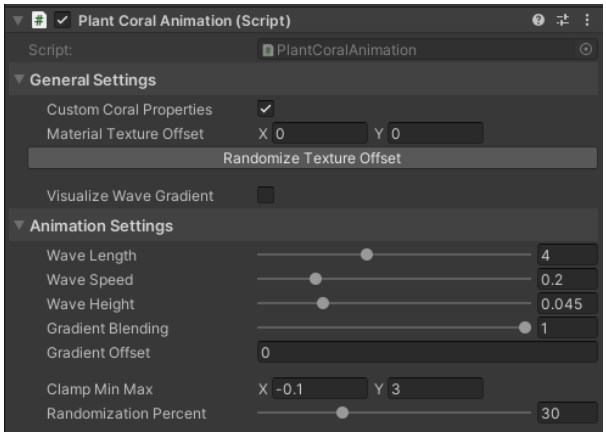
▼ **Custom Coral Properties**

- Enables the use of custom coloring options specifically intended for corals.

▼ **Material Texture Offset**

- Sets a custom texture offset value for that object instance in order to get a different color. Clicking the **Randomize Texture Offset** button will choose a random texture offset *(see bottom gif)*. Offset values are integers and increment by whole numbers, with a maximum value of 6 as the texture atlas is 6 tiles wide.

▼ **Visualize Wave Gradient**

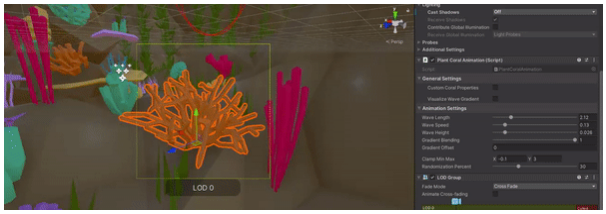


All serialized properties in the "Plant Coral Animation" component.

- Visualizes the wave gradient by changing the color of the mesh to a grayscale representation, with a black color signifying no distortion and a white color signifying full distortion.

▼ **Animation Settings**

- Contains properties which control the various aspects of the main distortion wave, including the wave gradient clamping and how randomized each instance of this object can be.



Changing the color of a coral object with the Material Texture Offset value.

UnderwaterEffect

The **UnderwaterEffect** script controls all underwater visual effects and the transition between surface and underwater effects. This includes post processing, fog, skybox tint, and refraction:

▼ Water Detection Offset

- The local offset at which UnderwaterEffect will detect if the camera is submerged.

▼ Effect Settings

▼ Underwater / Surface Profiles

- The **Surface** and **Underwater** post processing profiles respectively. What types of post processing profiles these are will depend on your render pipeline. These properties will not appear in the Built-in RP if the Post Processing package is not installed.

▼ Fog Color

- The color of the underwater fog effect.

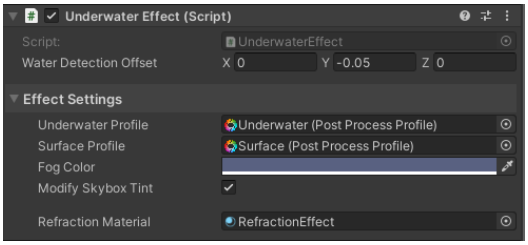
▼ Modify Skybox Tint

- Toggle to tint the skybox to the Fog Color value when underwater if your skybox shader contains a tint property.

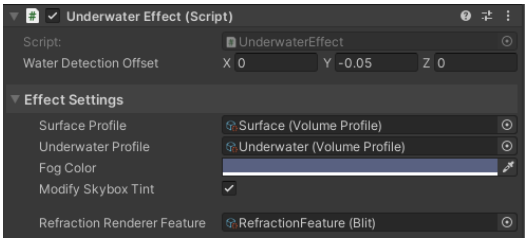
▼ Refraction

- Refraction is handled differently depending on which render pipeline you use *(see images)*.

▼ Built-in RP - Refraction Material



All serialized properties in the Built-in Render Pipeline version of the "Underwater Effect" component.



All serialized properties in the Universal Render Pipeline version of the "Underwater Effect" component.

- The material which uses the refraction post-processing effect shader. Unless you are using one of your own, this should always be **RefractionEffect**.

▼ URP - Refraction Renderer Feature

- The renderer feature which controls the refraction effect. This will be named **RefractionFeature** if using the Low Poly Underwater Pack's standard Universal Render Pipeline Asset. If you are using your own URP Asset, see the Universal Render Pipeline Asset Settings section to learn how to properly set up the refraction effect.

Caustics

The **Caustics** shader projects caustics onto all objects with the material below a certain level, taking light direction into consideration. The shader works by overlaying 2 different scrolling caustics textures and blending them together. This is the default shader in the Low Poly Underwater Pack:

▼ **Caustics Start Level**

- The level at which caustics will start, or the level of the water. This will have to be manually adjusted to whatever height your water is at.

▼ **Caustics Blending**

- The fade between the caustics start level and when the caustics fully appear.

▼ **Caustics Texture**

- The texture the caustics use to project onto the terrain.

▼ **Caustics Color**

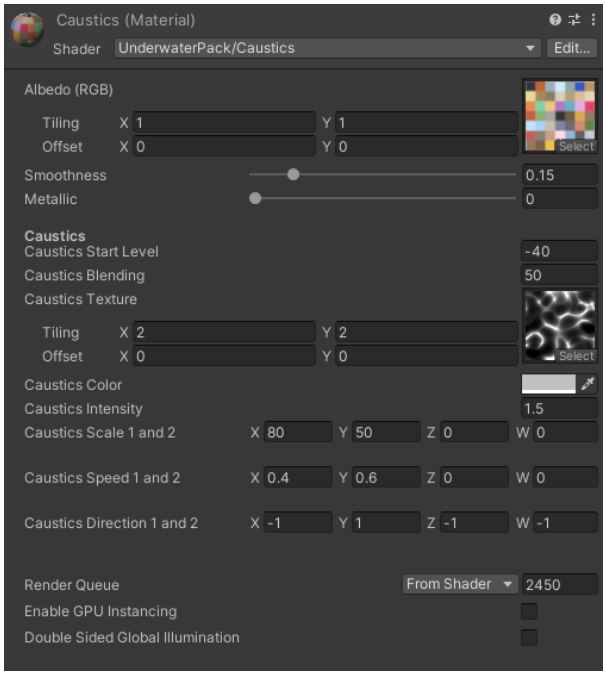
- The color of the caustics.

▼ **Caustics Intensity**

- How bright the caustics are.

▼ **Caustics Scale 1 and 2**

- The scale of the two caustics textures, with the x and y values representing them



All serialized properties in the "Caustics" shader.

respectively.

▼ **Caustics Speed 1 and 2**

- The speed of the two caustics textures, with the x and y values representing them respectively.

▼ **Caustics Direction 1 and 2**

- The direction of the two caustics textures, with the (x, y) and (z, w) values representing them respectively.

Note that, as said in the Forward vs Deferred Rendering Paths section, caustics will not react to light direction when the camera is using the deferred rendering path in the Built-in Render Pipeline.

Ships

The Low Poly Underwater Pack includes a number of ship prefabs *(Ventuar > UnderwaterPack > Prefabs > Manmade > Ships)* for your use. With the exception of the sunken ship, all ship objects contain customizable/removable sails, rigs, masts, doors, and wheel(s). Sails can be either raised or lowered, with the lowered versions being in the same heirarchy directory as the raised versions (typically as a child of the yard objects).

Each ship has their own use, features, and caveats to be aware of:

▼ **Dynamic Ship**

- A fully customizable ship prefab embellished with props and including buoyancy functionality. If you would like a fully furnished ship that interacts with and floats in the water, simply drag and drop this prefab into your scene.

The list of all ship prefabs available in the pack.

Dynamic ship example in the Demo scene.

▼ **Full Ship**

- A ship prefab with everything listed above except for buoyancy. Use this when you would like a fully furnished but static ship object.

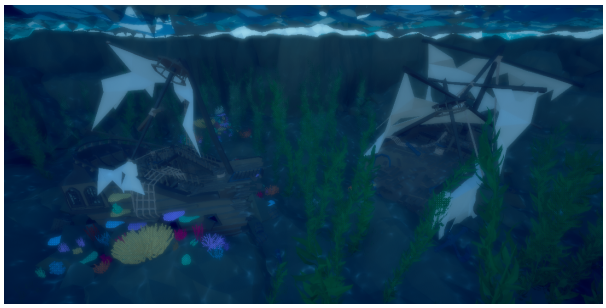
▼ **Ship Base**

- A base ship with no props inside and no buoyancy. Still includes customizable sails, rigs, masts, doors, and wheel. Use this if you

would like a static base ship object to customize yourself.

▼ **Sunken Ship**

- This ship is a completely static prop, with the only extra functionality being movable/removable captain's quarters doors *(see bottom picture)*.



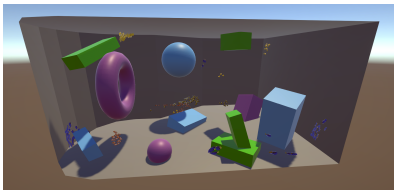
Sunken ship example in the Demo scene.

Boids Test Scene

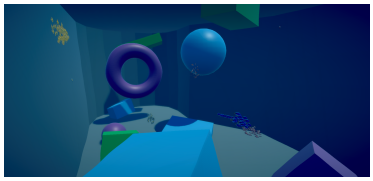
The **Boids Test** scene is a sandbox where you can try new AI settings, see how different fish react with obstacles, or test limitations of the boids AI by seeing just how many fish you can spawn and still maintain a reasonable framerate (my favorite 😄).

Before using large quantities of boids in your own scenes, I would highly reccomend testing them out in this scene to see how well your hardware can handle them and if there are any optimizations/changes that could be made to better suit your needs. Treat this as a dedicated space to play around with and configure the fish to your heart's content!

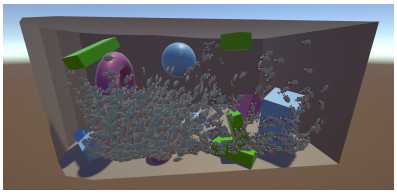
In this scene is a camera, directional light, spawner object with a Boid Master component attached, and the Boids Test Room. Use the spawner to control the type and amount of fish that spawn. Additionally, there is special logic to activate underwater effects when going inside the Boids Test Room for aesthetic purposes *(see middle picture)*.



Outside the Boids Test Room.



Inside the Boids Test Room with the underwater effects.



These are 2500 pirhanas—perhaps a bit too much! 😬 At this quantity and for my hardware, they start going through the walls and out of bounds when the framerate dips too low.

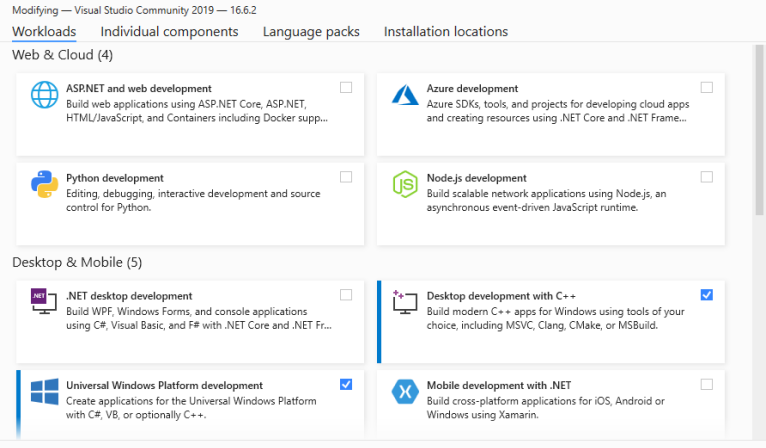
> 💡 Note that larger single fish such as the sharks, Orca, etc. may not work too well in the Boids Test scene as it is due to reasons I mentioned in the "Single Fish" section. Feel free to increase the scale of the Boids Test Room object to fix this.

Compilation

Extra steps may need to be taken when compiling projects using this pack. These issues typically involve either the use of global forgot issues with the Burst Compiler. If any other issues arise when compiling, contact me and I will at the very least try to point you in the right direction.

▼ Burst Compiler

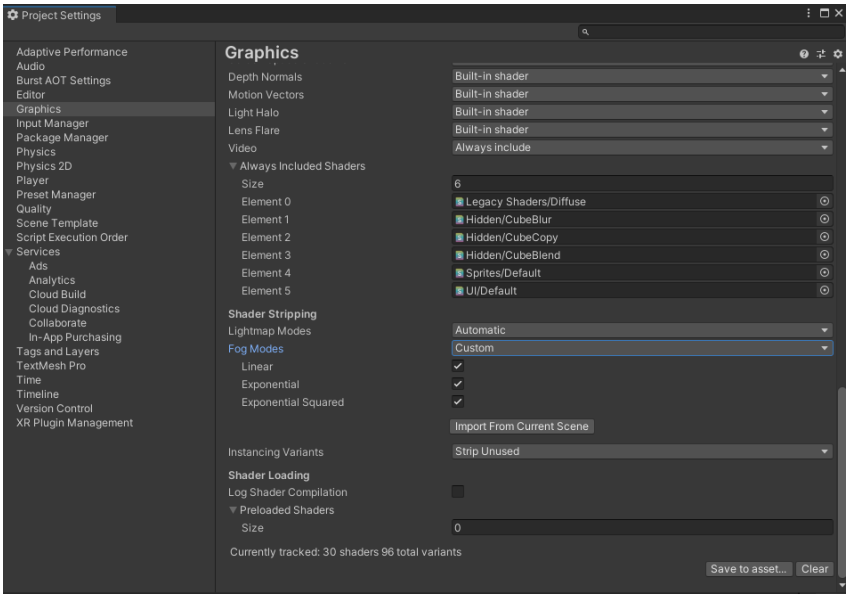
- Problems may arise while compiling projects with the Burst Compiler if Visual Studio is being used or has been installed on your computer.
- If this is the case, navigate to the *Visual Studio Installer* > your in-use version of Visual Studio, select *Modify*, and ensure that "Desktop development with C++" and "Universal Windows Platform development" are selected.



An example of what your Visual Studio version "Modifying" page should look like.

▼ Fog

- Go to *Edit* > *Project Settings* > *Graphics* > *Shader Stripping* > *Fog Modes* > *Custom*, and ensure all checkmarks are ticked before compiling.



An example of what your project's Graphics settings should look like with all boxes for "Custom" fog mode checked.

Optimization Tips

Overusing elements of the Low Poly Underwater pack can lead to heavy hits in performance if not used carefully. There are some features which can be especially taxing on performance:

▼ **Fish**

- Large numbers of fish, especially when each has a high-quality collision detection configuration, can oftentimes be a significant culprit of performance decreases. Oftentimes, the performance slows down moreso due to the collision detection the fish perform than the sheer quanitity of fish in the scene, especially with a well-configured disable distance set in Boid Master.
- Use the BoidsTest scene to see how different fish settings/quantities can impact your performance.

▼ **Buoyant props**

- Especially when buoyant props are using accurate water detection (set in BuoyancyMaster), they can be taxing when computing in larger than small-to-moderate quantities.

▼ **Grass**

- The grass in the Low Poly Underwater Pack are simply meshes, which is an admittedly rather inefficient method of handling grass—especially in large quanities.

If performance ever becomes too slow in the Demo scene or any scene which you use the Low Poly Asset Pack, there are a number of ways you can try to improve it:

▼ **Using the C# Jobs boids algorithm**

- The C# Jobs boids algorithm is much more performant than the alternative algorithm. If at all possible, install the Burst package and

▼ **Decreasing the amount of fish in the scene**

- Boid and collision detection calculations, even at their most performant settings, can often be a significant culprit of poor performance in large quantities.

use this method to improve boid performance.

▼ **Decreasing "Boid Disable Distance" in BoidMaster**

- Doing so will oftentimes decrease the number of boids that perform calculations at any given moment in your scene, but could result in boids popping in and out of being culled if set too low.

▼ **Toggling off "Use Accurate Detection" in BuoyancyMaster**

- The "Use Accurate Detection" toggle in BuoyancyMaster can be used to achieve more accurate water detection in buoyant objects, but can also lead to considerable decreases in performance. I reccomend toggling this off if you will not be viewing buoyant objects up-close in your scene.

▼ **Decreasing "Accurate Buoyancy Dist" in BuoyancyMaster**

- By decreasing the "Accurate Buoyancy Dist" value, you decrease the radius around you in which buoyant objects use accurate water detection, resulting in the objects closest to you using accurate detection, but ones farther away will approximate to save on performance.

▼ **Lighting/shadow settings**

- This isn't necessarily specific to the Low Poly Underwater Pack, but tweaking shadow and lighting settings can always alter performance. Keep in mind that most underwater props (corals, plants, rocks, etc.) do not use shadows, so shadows should not be a significant culprit of poor performance (at least in the Demo scene).

▼ **Decreasing the amount of buoyant objects/float points in the scene**

- Buoyant objects, regardless of water detection methods, can significantly decrease performance if used in large quanitites. Either decrease the amount of buoyant objects or decrease the float points of those objects to save on performance.

▼ **Reducing the amount of grass in the scene**

- As mentioned above, the way grass is handled in the pack is rather inefficient. Grass LOD cull distances are set rather low to combat this, but reducing the amount of grass in your scene can help save on performance.

▼ **Decreasing boid collision detection settings**

- As mentioned above, performance oftentimes slows down due to collision detection calculations moreso than anything else the fish do. To decrease boid detection accuracy and increase performance, modify these values in either **SingleFishAI** or **FlockingBoidAI**:
  - Decrease "Detection Quality"
  - Decrease "View Angle"
  - Increase "Turn Fraction"
  - Decrease "Accuracy"
- Keep in mind that **increasing** the "turn fraction" value will result in **lower-quality collision detection**, contrary to the other 3 values.

▼ **Tweaking LOD cull distance**

- LOD cull distances should already be rather low (fine-tuned to the Demo scene to maximize performance), but tweaking these values to be even low can save on rendering costs.

Feel free to experiment with any other settings I may have missed or you feel may improve performance! If I missed something on this list, please contact me so I can include it.

Known Issues

If you viewing the web version, this area will be updated with more information as new bugs are discovered in the pack. If you are viewing the PDF version, this area may not contain all known issues until the pack is updated.

- Boids sometimes tend to align in a straight line, resulting in them clipping into each other and giving an undesirable look to the flock. This can often be seen in the Boids Test scene, and can sometimes be mitigated by decreasing cohesion force.
- Foam can sometimes behave unrealistically, with the water oscillating between containing no foam and later containing lots of it, if the "Foam Distortion Amount" value of WaterMesh is set low enough.

Conclusion

If you've read until here, thank you so much! It was certainly a lot to read, and this by no means needs to read this in its entirety, but I appreciate you taking the time to use this documentation page. I truly hope that you'll be satisfied with everything that is offered in this pack, and that your game or project will be elevated because of it. As I said in the introduction, I am proud of how this pack came out and the work that's been put into it, and I hope it will not disappoint.

I will continue to update this documentation with relevant information as long as updates/patches are being made. If there is something missing, incorrect, or vauge in this page, feel free to leave a comment or contact me directly using my contact information below.

Once again, if you enjoy the pack, it would be extremely appreciated if you left a positive review in the Unity Asset Store or CGTrader. If you wish to contact me for any reason, please don't hesistate to do so. I am currently a full-time student but I will try my absolute best to get back to you in a prompt manner. Thank you! 🐟🌊

▼ **Contact Information:**

- **Email** - ventuargames@gmail.com
- **Twitter** - @Ventuargames

Table of Contents