

Zibra Liquids 1.4.0, plugin documentation

ZIBRA LIQUIDS IS A PLUGIN FOR THE UNITY ENGINE. IT ALLOWS THE USE OF REAL-TIME FLUID SIMULATION (GPU) POWERED BY AI BASED OBJECT APPROXIMATION.

Features	Zibra Liquids free	Zibra Liquids
Mobile support (iOS)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Particle count limit	2 mln	10 mln
Analytic colliders (simple shapes like cubes, spheres and bowls)	up to 5	unlimited
Voxel colliders (arbitrary geometry as collider)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Collider friction	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Force interaction setting	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Liquid emitters	1	unlimited
Liquid voids	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Liquid force fields	Radial (only one instance)	Radial, Directional, Swirl (unlimited)
Liquid initial state baking	<input type="checkbox"/>	<input checked="" type="checkbox"/>



Table of contents:

1	Create Zibra Liquid
2	Configure Zibra Liquid
3	Colliders
4	Manipulators
5	Liquid Initial State Baking (experimental)
6	Registering Zibra Liquids
7	Troubleshooting

System Requirements

Editor platforms: PC (DX11, x86 and x64), macOS(Metal, x64 and arm64)

For M1 Mac users, the Apple Silicon version of Unity Editor is strongly recommended.

Build platforms: PC(DX11, x86 and x64), UWP(DX11, x86 and x64), macOS(Metal, x64 and arm64), iOS (iOS version 12 or later, iPhone 6s or newer)

Unity version: 2019.4 or later (use the latest feature update)

VR is unsupported

Supported Render Pipelines: SRP, URP, HDRP

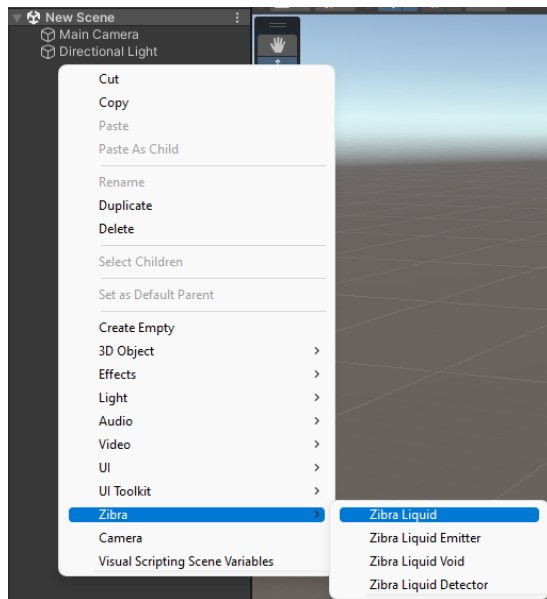
URP: 7.5 or later

HDRP: 7.x or later

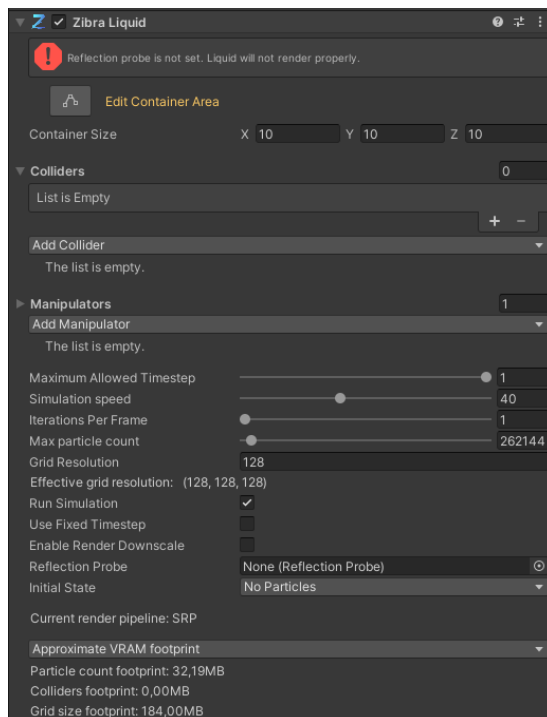
Create Zibra Liquid

To create a Zibra Liquid instance:

1. Open the Hierarchy window
2. Right click where you want to add Zibra Liquid instance
3. Click on the “Zibra -> Zibra Liquid” option.



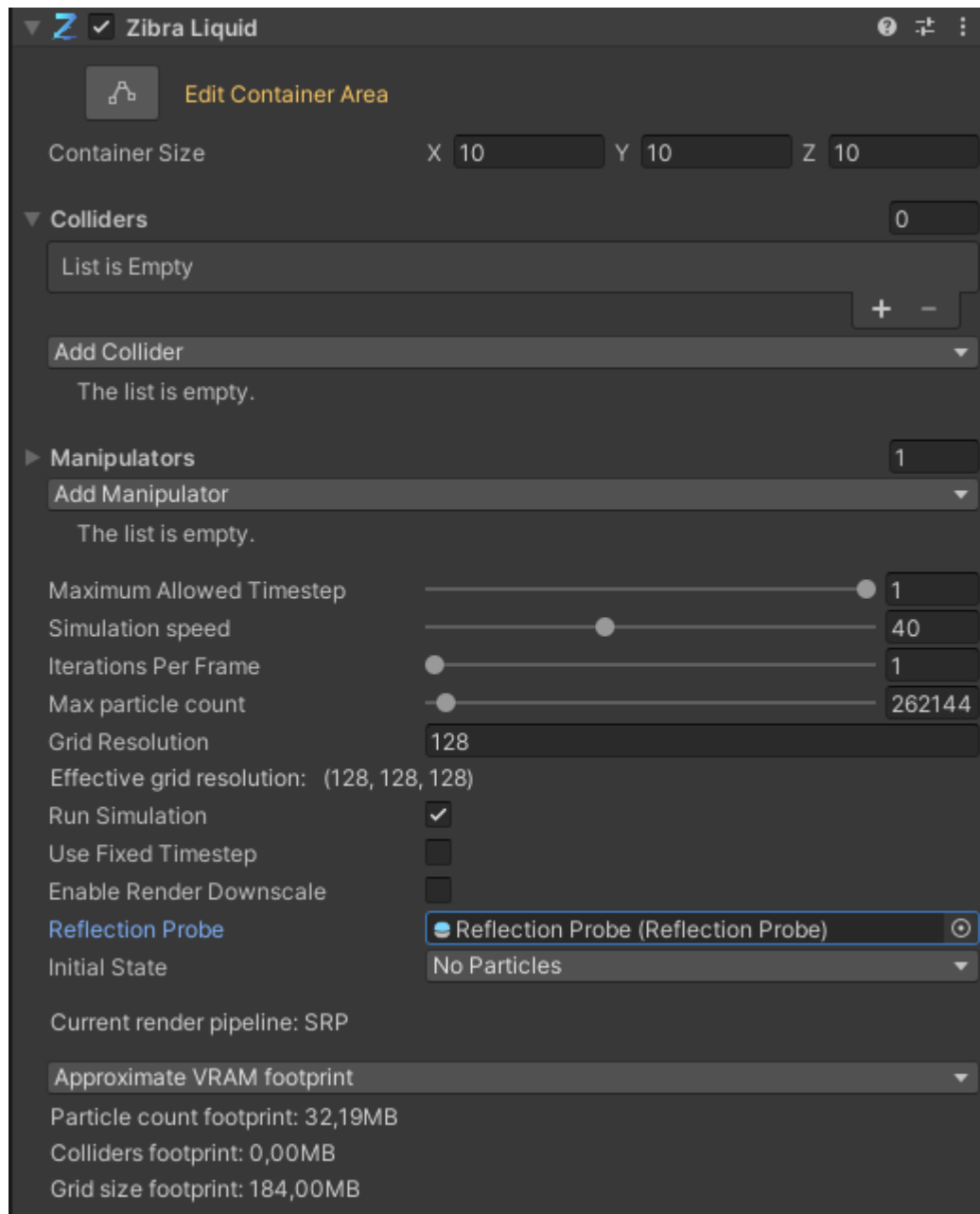
After that, In the Inspector window, you'll see the Zibra Liquid parameters:



After that you need to set the “Reflection Probe” parameter. On SRP/URP it’s strongly recommended that you set this parameter, but is not strictly necessary. On HDRP you must set a reflection probe before you can use the liquid instance.

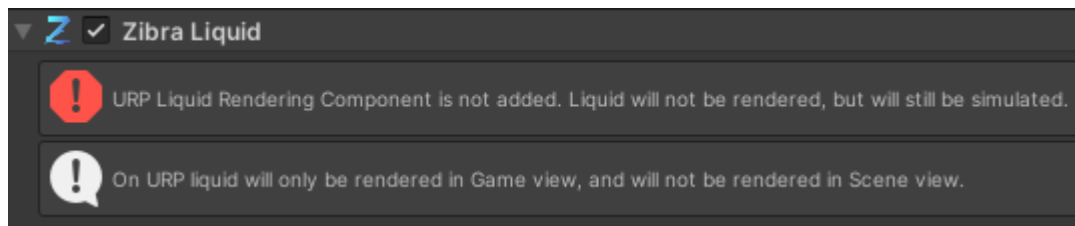
You can check which render pipeline you are using by looking at the “Current render pipeline” message - on the screenshot the SRP is shown.

On SRP, after setting the reflection probe, you’ll no longer see any errors/warnings.

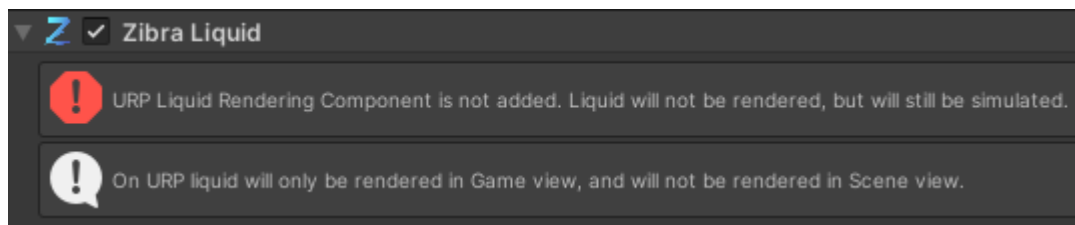


Additional setup on URP

If you are using URP you'll see those messages:

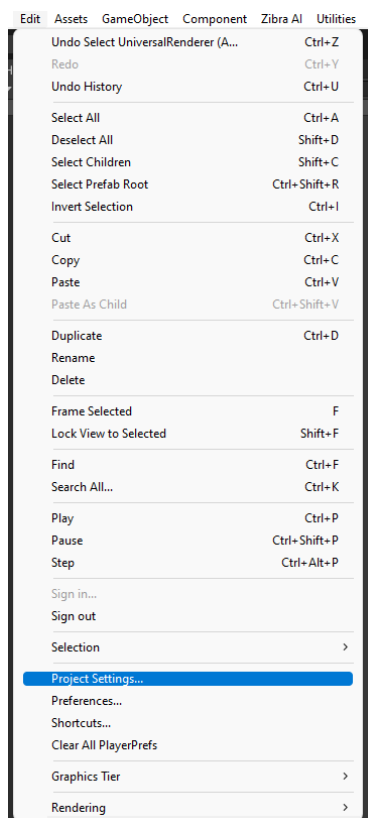


If you are using URP you'll see these messages:

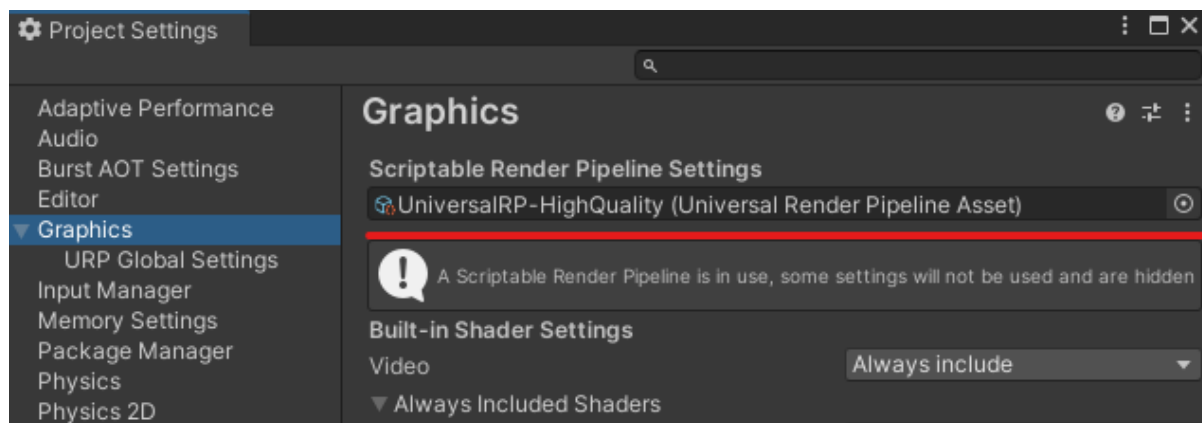


You need to add “*URP Liquid Rendering Component*” to your Universal Renderer asset. You can find your Universal Renderer asset by following this steps:

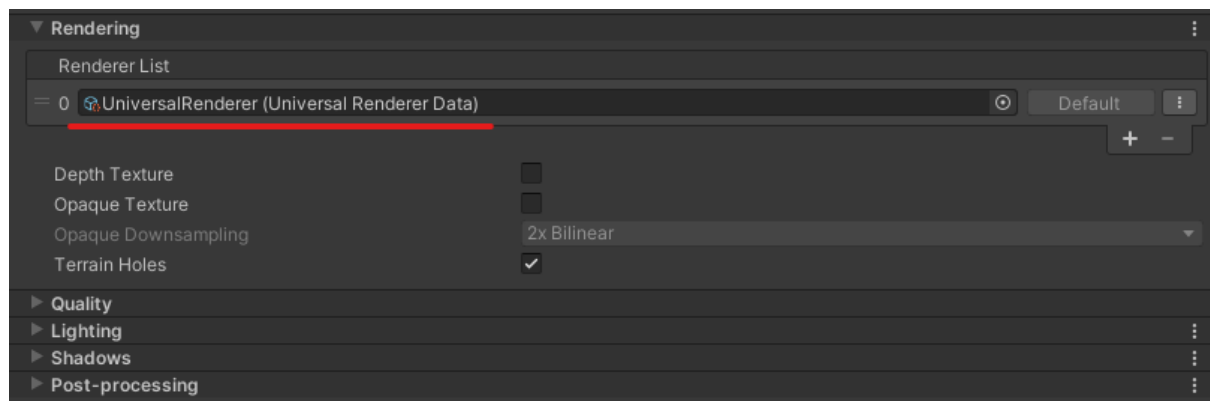
1. Navigate to “*Edit -> Project Settings...*”



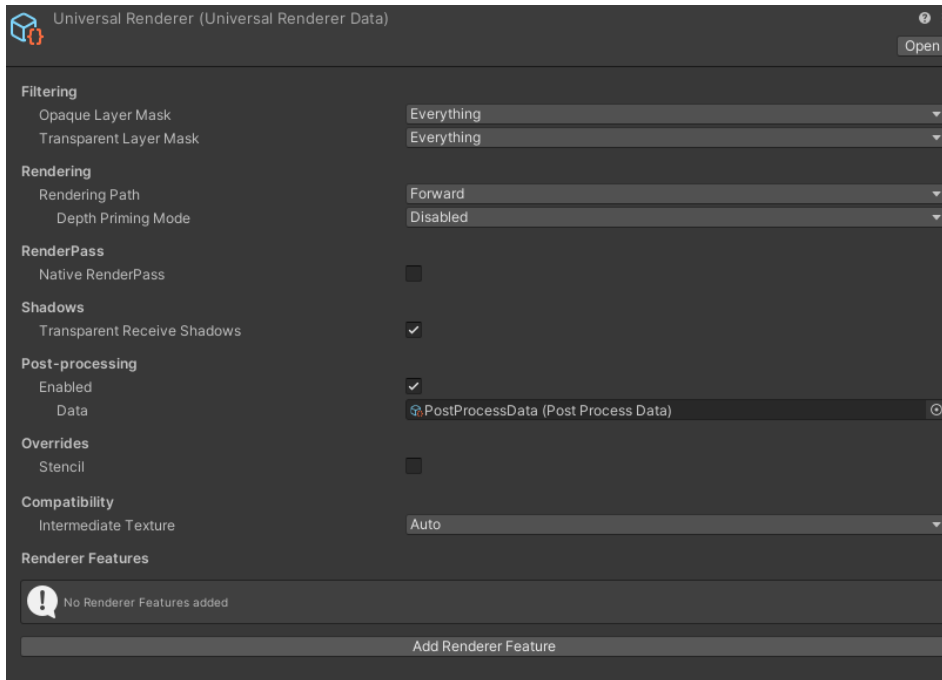
2. From there go to Graphics



3. Scriptable Render Pipeline Settings is your current Universal Render Pipeline Asset. Open that asset in Inspector (*you can do it by double clicking it in project settings*) You'll be able to see your Universal Renderer there

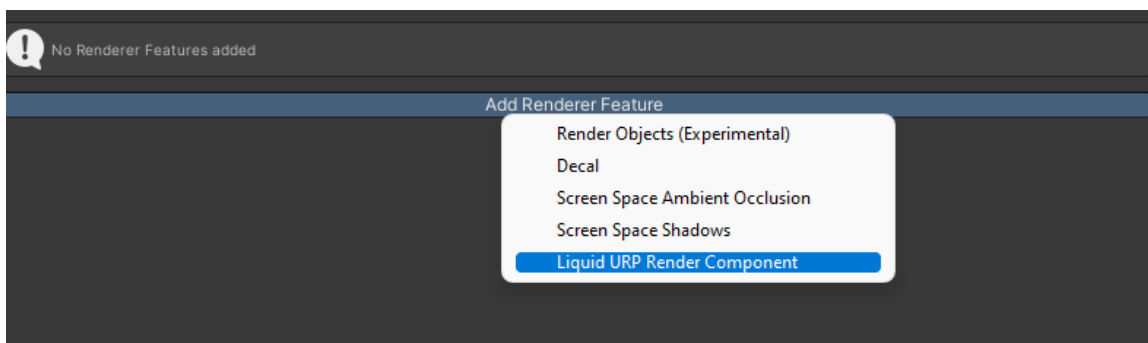


4. Open it in the Inspector (by double clicking)

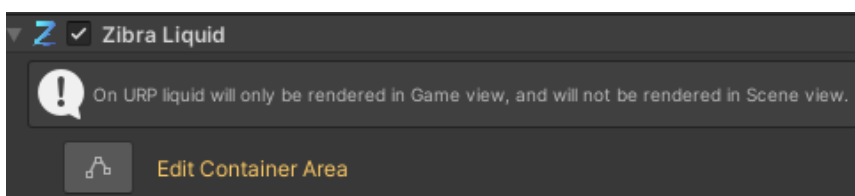


(Specific UI elements can vary from version to version)

From there you can finally add “URP Liquid Rendering Component”:



If you did everything correctly, error in Zibra Liquid will disappear:

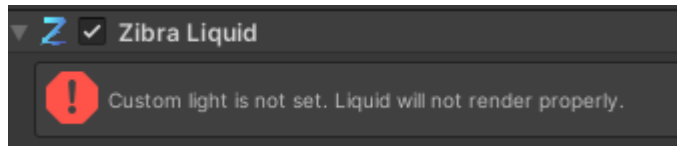


Note that adding “URP Liquid Rendering Component” is project wide, and is only needed once.

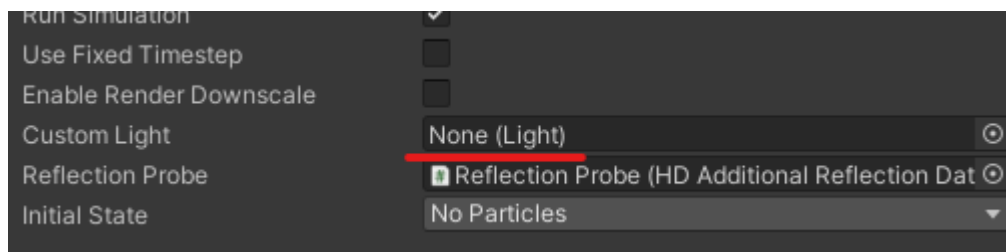
Lastly: on URP, liquid will only be rendered in Game view, including the baking utility.

Additional setup on HDRP

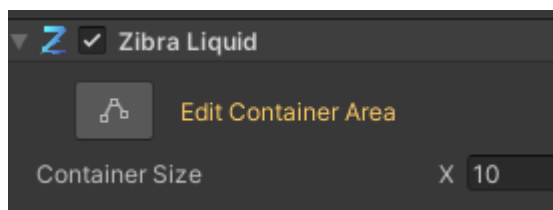
If you are using HDRP you'll see this message:



On HDRP you also need to set the Custom Light parameter. This parameter sets which light will be used for liquid lighting. Currently, only 1 light can be used for lighting the liquid.

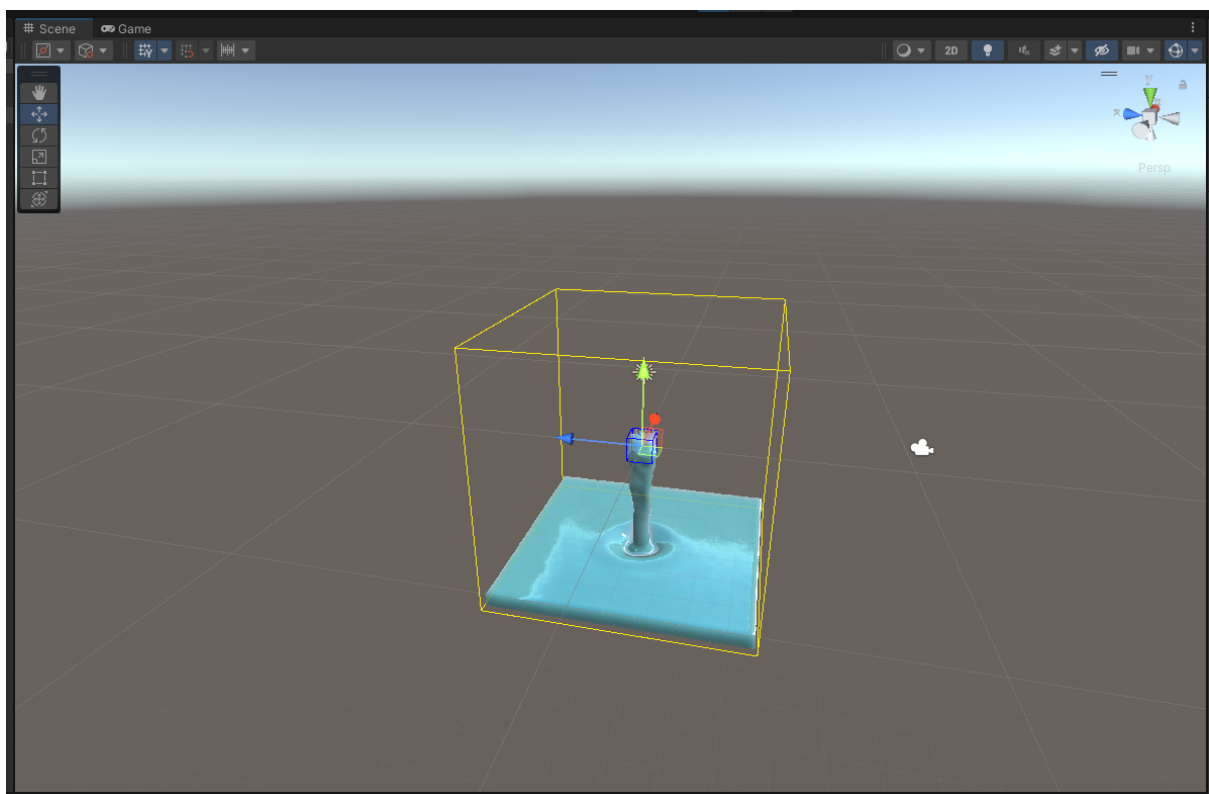
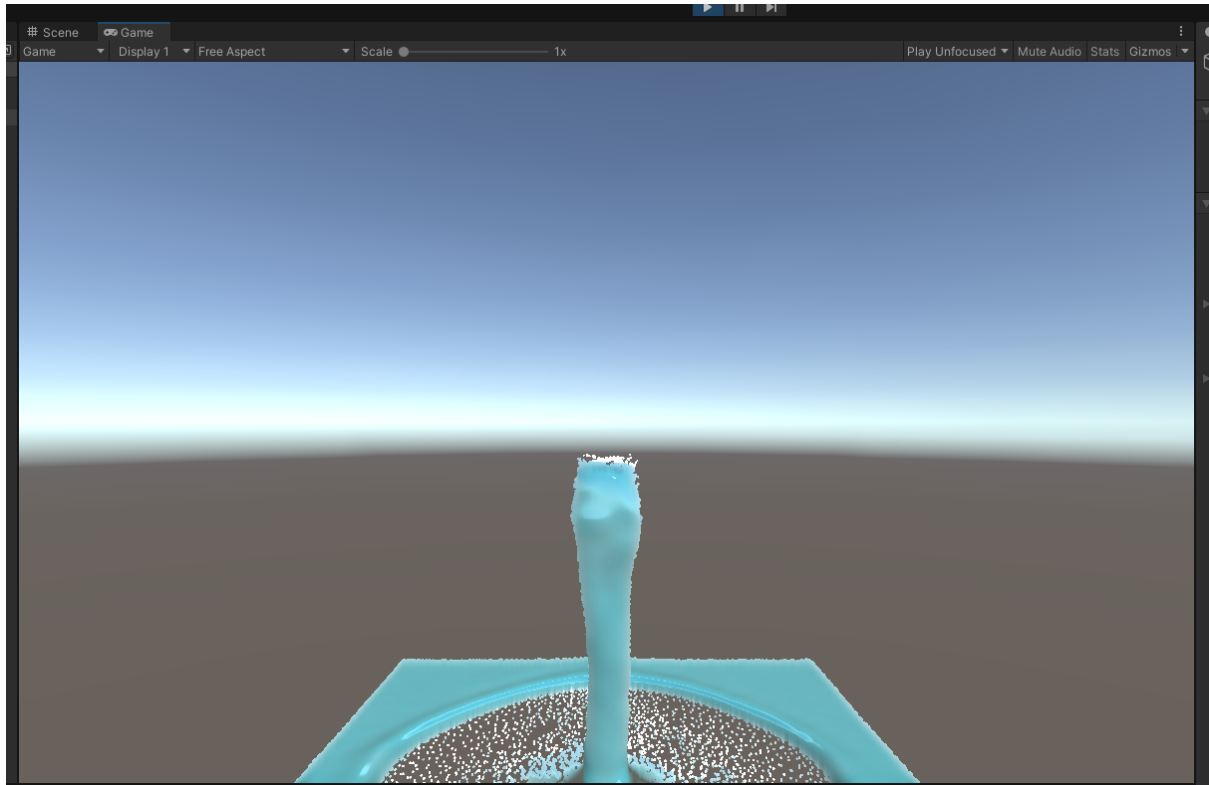


After setting the Custom Light parameter the error message will disappear:



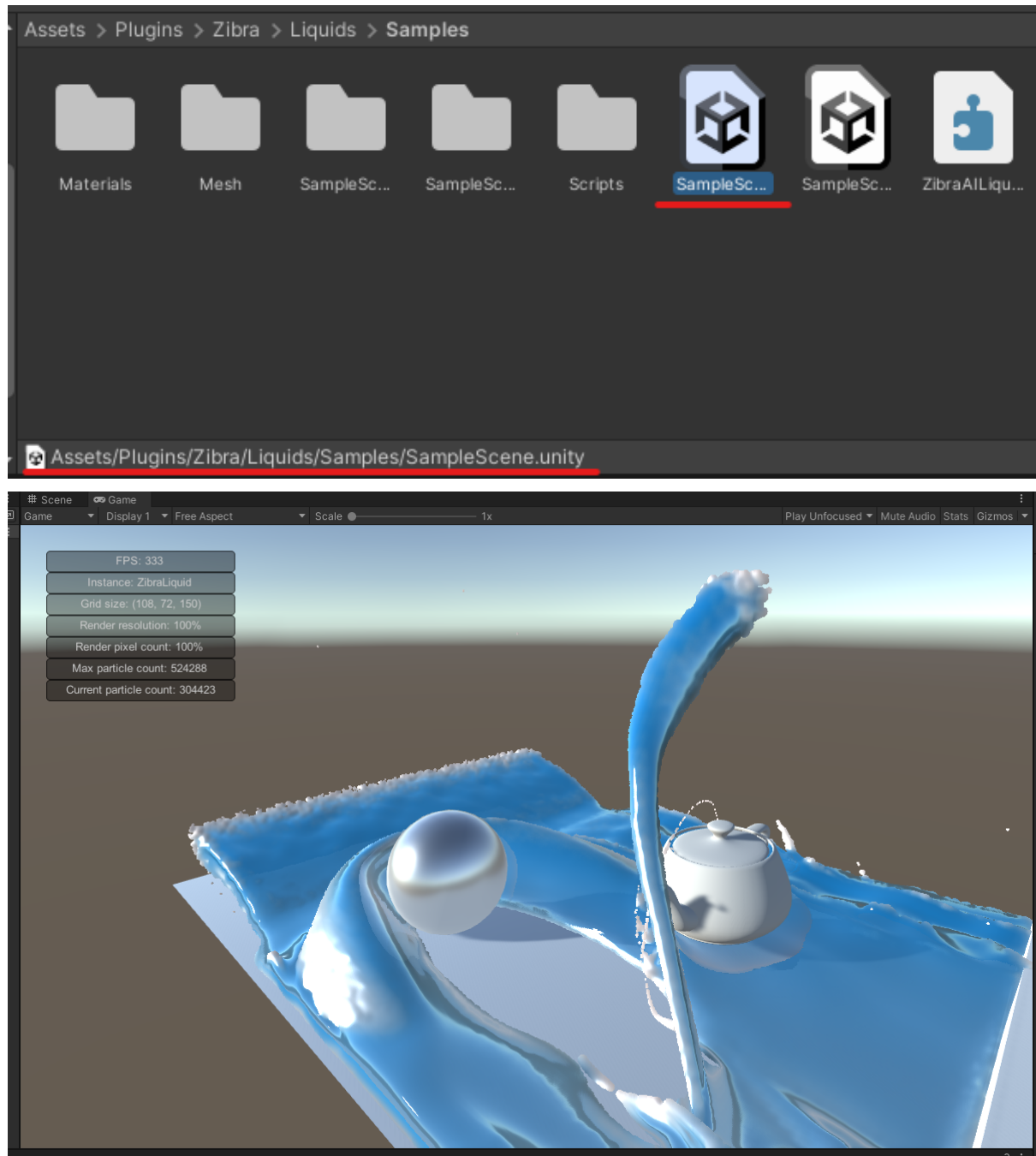
After creating Zibra Liquid

You can now press play, and see liquid (in URP you can only see liquid in Game View, on SRP/HDRP you can see liquid in both the Scene and Game Views)



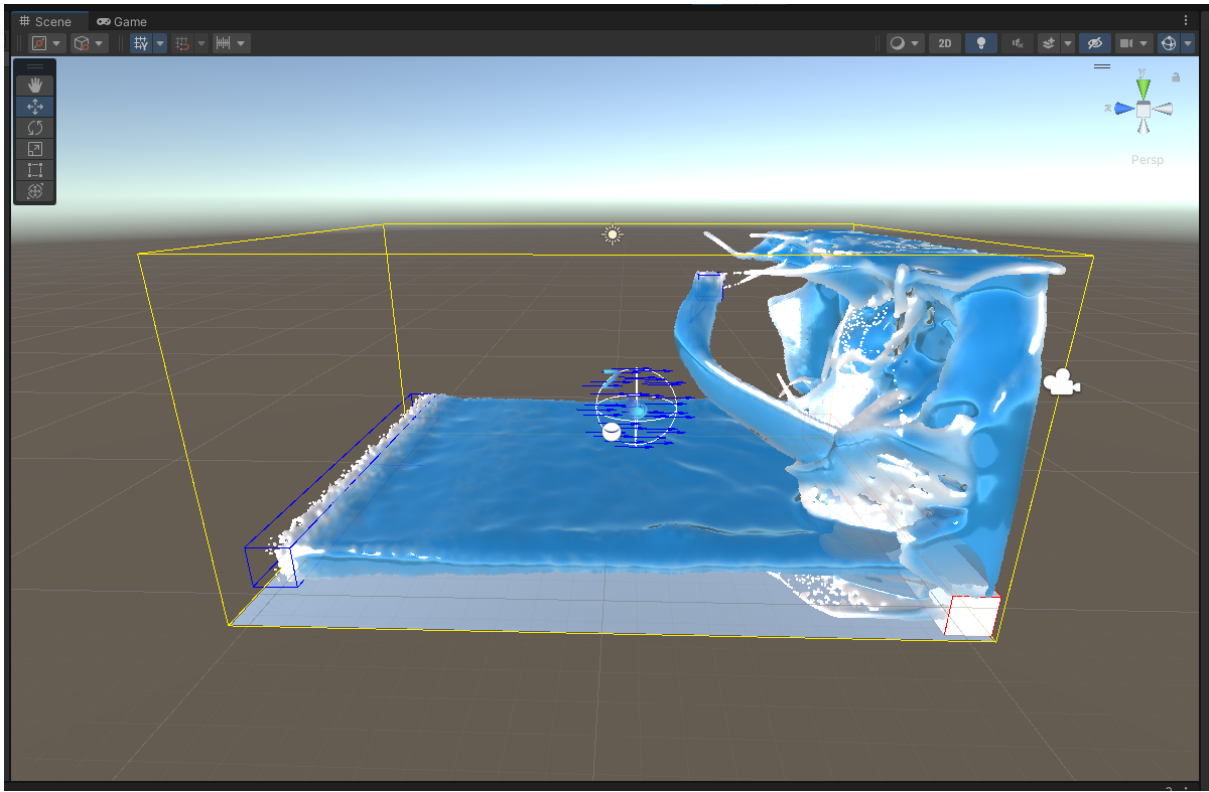
Also, you can check out how the fluid works in the built-in demo scene. To do that, open the Project window and go to *Assets -> Plugins -> Zibra -> Liquids -> Samples* and open the *SampleScene*.

Note that the sample scene is for SRP, and in URP and HDRP non liquid objects may look incorrectly.



In the sample scene you can see various features of Zibra Liquid.

You can move the camera with *WASD*, control gravity with arrows, remove gravity with *O*, increase and decrease gravity with *Shift* and *Ctrl*

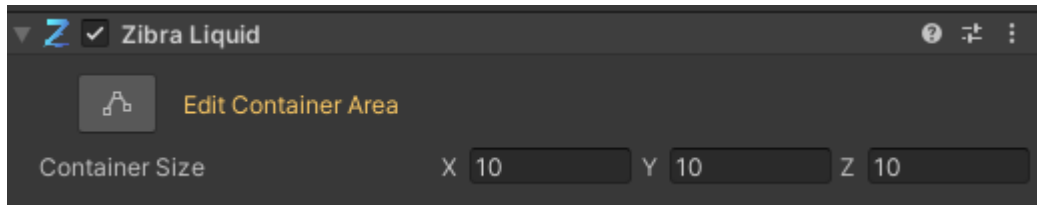


Configure Zibra Liquid

Liquid parameters

Main parameters

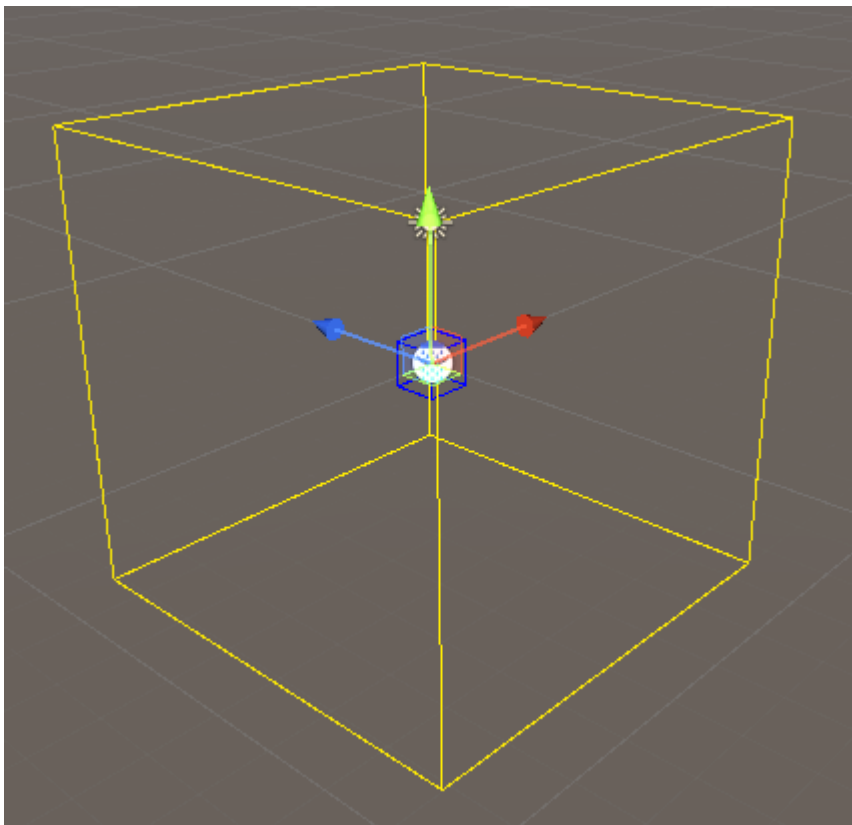
- **Container size**

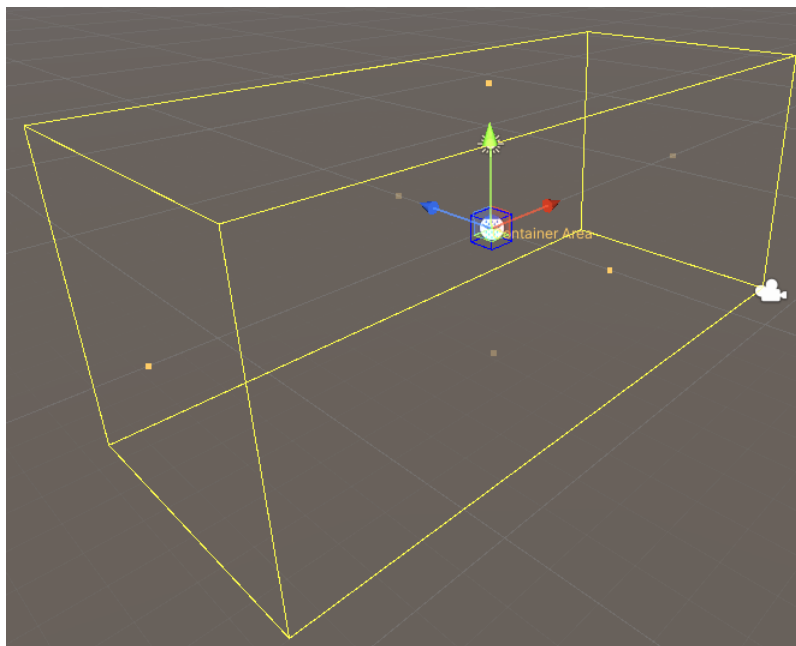
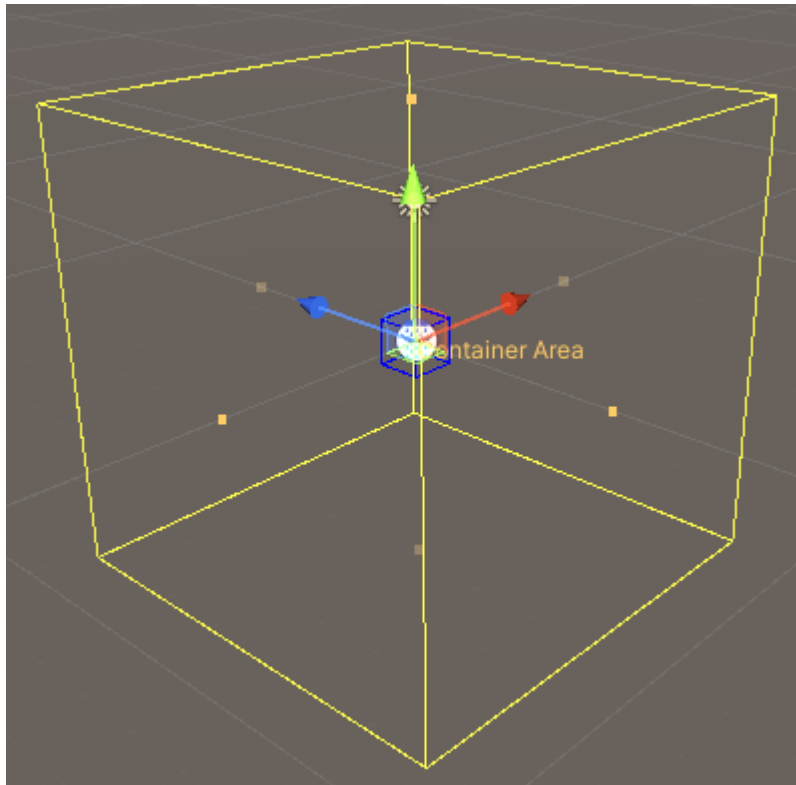


The Container size parameter changes how large the simulation volume of liquid is.

You can press “Edit Container Area” to enable resizing gizmos.

This parameter cannot be changed on “live” liquid instance.





- **Maximum allowed timestep**



This parameter specifies what the highest delta time that may be used in simulation is, when the FPS drops. If that limit is hit, liquid simulation will slow down.

Higher values correspond to less stable simulation during FPS drops, but will result in less cases when the liquid noticeably slows down.

- **Simulation speed**

Simulation speed  40

This parameter controls the speed of the liquid simulation. You can slow down or speed up the liquid with this parameter.

- **Iterations per frame**

Iterations Per Frame  1

Controls how many physic steps will be calculated each time. This severely affects performance, but results in slightly better simulation quality. In most cases you will want to set it to 1.

- **Max particle count**

Max particle count  262144

Controls how many liquid particles a liquid instance can have at the same time. Higher values correspond to a higher possible liquid volume, but results in higher VRAM usage and a higher performance cost.
This parameter cannot be changed on the “live” liquid instance.

- **Grid resolution**

Grid Resolution 128
Effective grid resolution: (128, 55, 55)

This parameter controls the size of the liquid simulation grid. Higher size of the grid corresponds to a higher quality simulation, and the smaller size of each particle, but results in higher VRAM usage and a higher performance cost.
Effective grid resolution shows you the 3D size of your liquid grid. VRAM usage and performance cost scales with effective grid resolution. This parameter cannot be changed on the “live” liquid instance.

- **Run simulation**

Run Simulation ☒

Freezes the liquid when disabled. Also decreases performance cost when disabled, since we don’t do physics simulation.

- **Use Fixed Timestep**

Use Fixed Timestep ☐

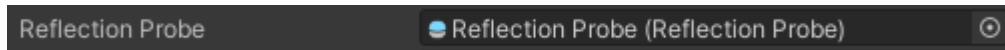
Use Fixed Timestep for physics simulation. Can be used to have almost identical behavior of liquid between different runs and machines. Use with care, if the physics simulation takes more time than Fixed Timestep, performance will be severely affected.

- **Enable Render Downscale**



Enables the usage of lower resolution for liquid rendering. The Resolution scale is controlled with the “Downscale Factor”. Recommended for mobile devices.

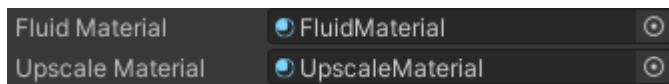
- **Reflection probe**



Sets reflection probe used for reflection calculation on liquid. Must be set on HDRP, and it's strongly recommended to be set on SRP/URP.

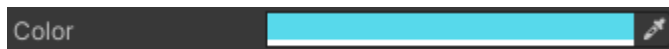
Material parameters

- **Materials**



Materials used to shade liquid surface, and optionally upscale liquid when Enable Render Downscale is enabled. Most users will not change that option. To use it you will need to create your custom version of FluidMaterial, and potentially UpscaleMaterial as well. If it is set to None in Editor it will revert back to the default value.

- **Color**



Liquid color.

- **Reflection Color**



Tint for reflections on the water.

- **Roughness**



How rough is the liquid surface. Normally, for liquids that parameter is really low.

- **Metalness**



How metallic (reflective) the liquid surface is.

- **Opacity**

How opaque the liquid is. Note that opacity is not the same for the whole liquid. In the same Zibra Liquid instance, the opacity of a shallow puddle and a deep lake will be different, just the same as it is in real liquid.

- **Refraction distortion**

Determines how much light is distorted as it passes through the liquid.

- **Particle scale**

Scale for particle rendering. Lower values correspond to more detail, but more artifacts. Needs to be at least ~1.5 to achieve a believable render. If set to very low values you'll see each individual particle. Does not affect simulation, it only affects rendering.

- **Foam parameters**

Configure what is the threshold for foam appearance and how intensive it is.

- **Blur radius**

Configures how strong the particle blur is. Lower values result in more detail, but the individual particles become more distinct.

Solver parameters

- **Gravity**

 X Y Z

Gravity that affects the entire liquid.

- **Fluid stiffness**

Configures how stiff the liquid appears (how resistant it is to compression). Higher values are not recommended.

- **Particle density**

Particle Density 1

Configures resting particle density. Higher values result in smaller particles, which result in higher quality, but smaller liquid volume.

- **Velocity limits**

Maximum Velocity 3
Minimum Velocity 0

Limits the maximum/minimum velocity of the particles. Setting minimum velocity to values higher than 0 results in liquid that will never stop.

- **Viscosity**

Viscosity 0

Configures how viscous the liquid is (how hard it is to change the shape of the liquid)

- **Surface tension**

Surface Tension 0

Configures the surface tension of the liquid. Higher values result in liquid that tries to merge together. Negative values result in liquid that tries to split into as many small liquid parts as possible, resulting in a spagettification effect.

- **Force interaction strength**

Force Interaction Strength 0

Scales the force that the liquid applies to objects with force interaction enabled. This has a logarithmic scale.

Colliders

Zibra Liquid has 2 types of colliders: Analytic and Voxel colliders.

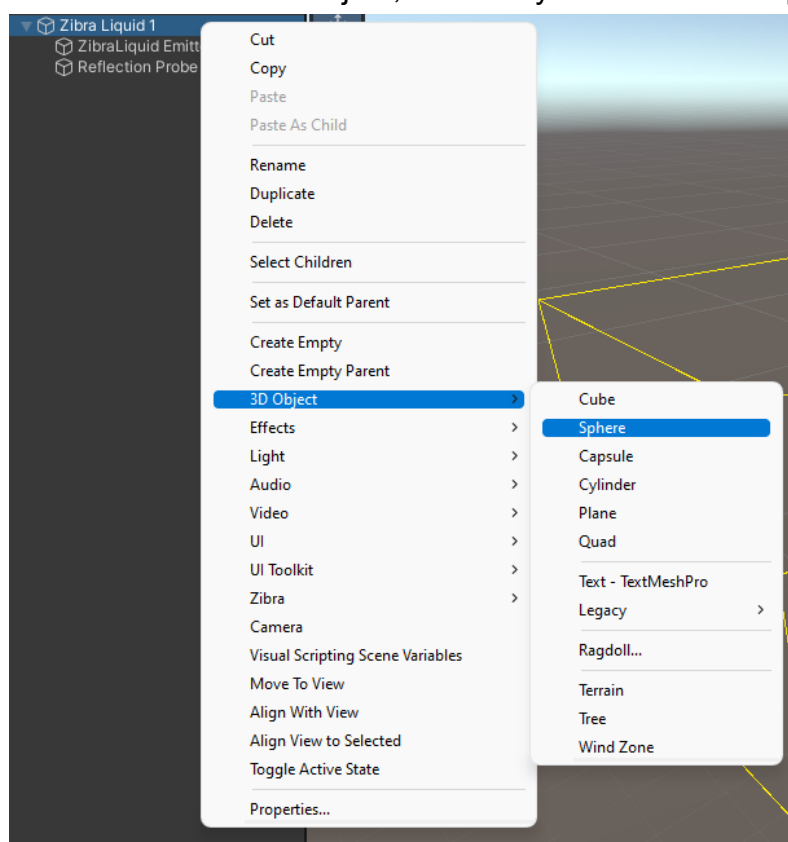
Analytic colliders are simple shapes: cubes, spheres, capsules, torus, cylinders.

Voxel colliders are created from arbitrary mesh.

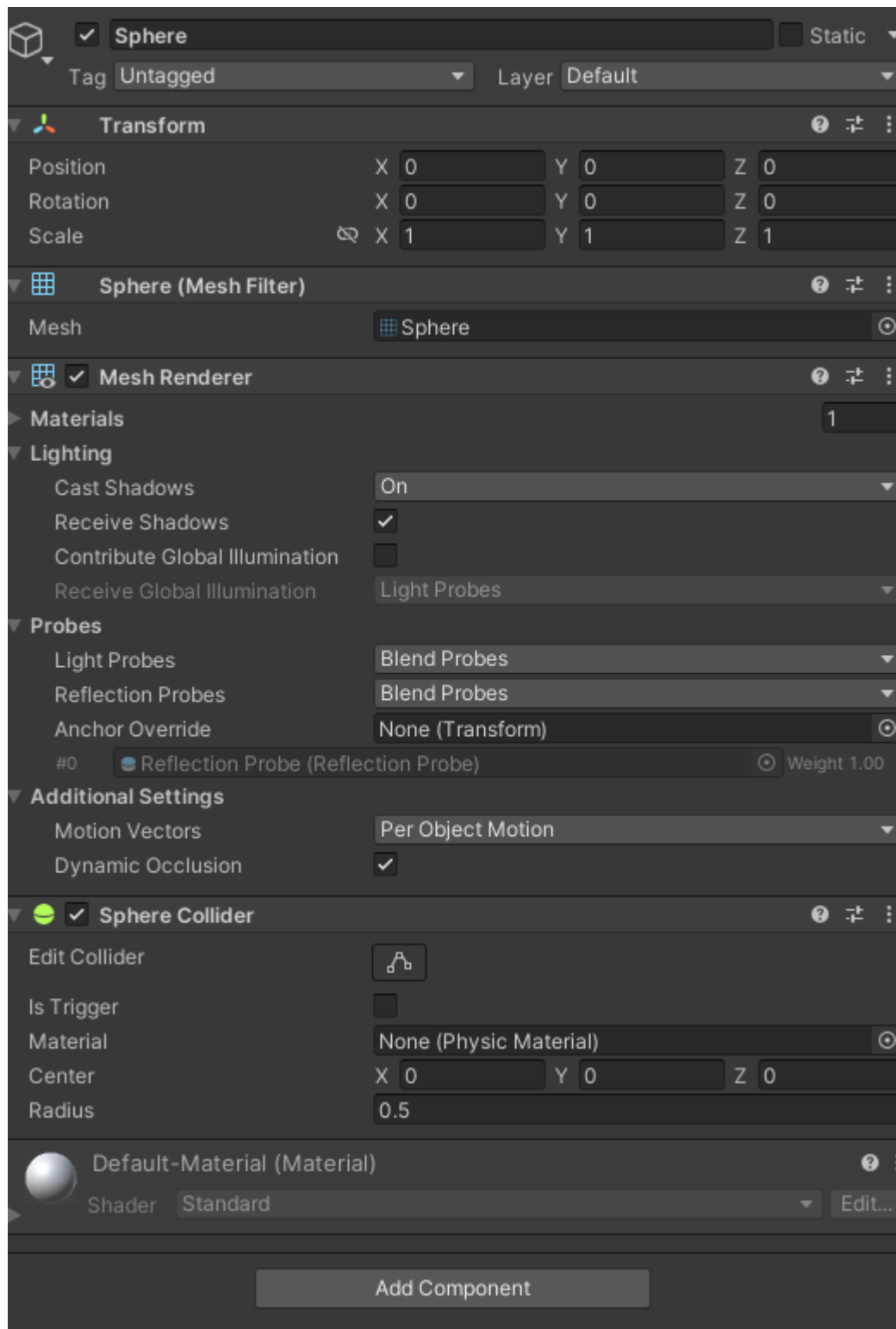
Adding Analytic Colliders

To add analytic collider:

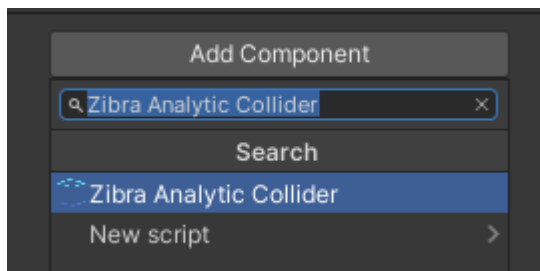
1. Create a new GameObject, either of your selected shape or empty one:



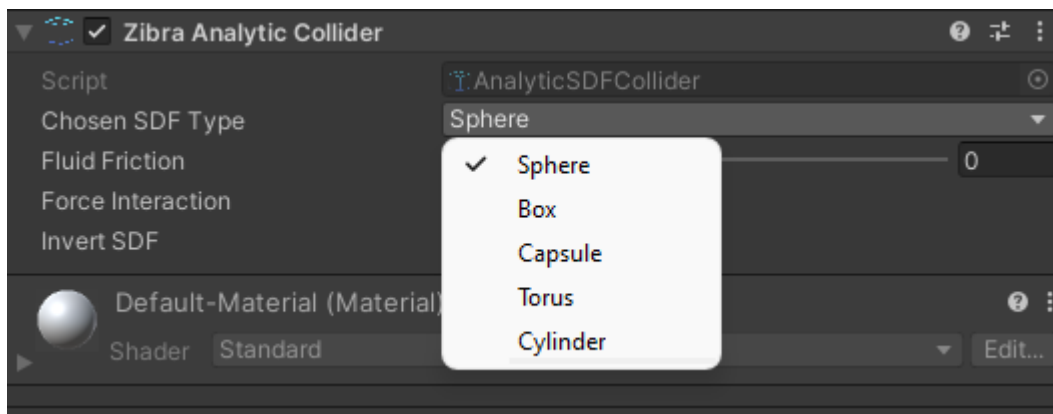
2. Open Inspector for newly created GameObject



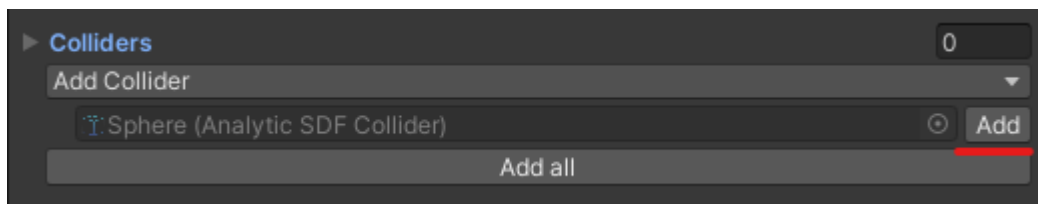
3. Press add component, and add “Zibra Analytic Collider”



4. Select correct “Chosen SDF Type”

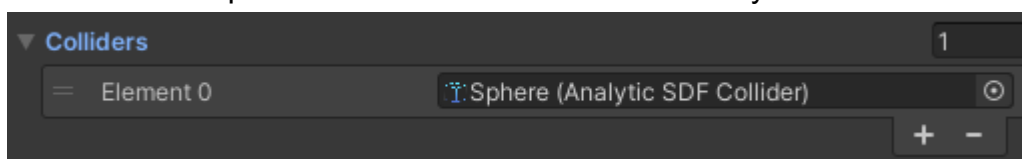


5. Open the Inspector window for Zibra Liquid and Press the Add button for your newly created collider.

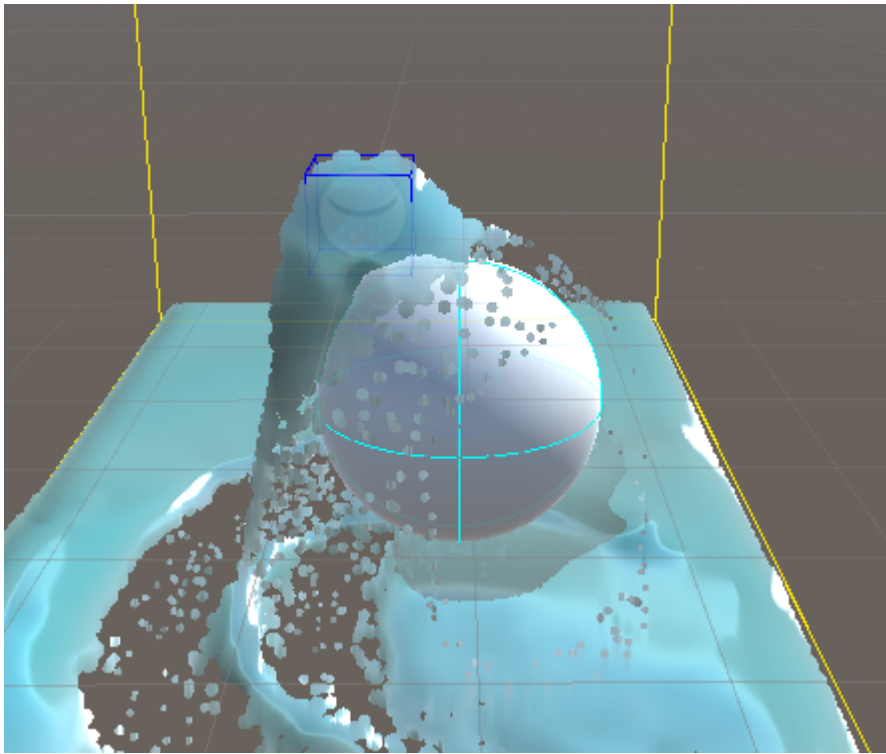


Note that you may have multiple liquid instances within a single scene, with a separate list of colliders. So you don't necessarily need to add all of the colliders to your liquid instance if you have more than 1 liquid instance.

6. You can now expand the full list of Colliders and see your colliders there



After that your collider is ready to be used. You can set it's transform in the editor, or move it at runtime.



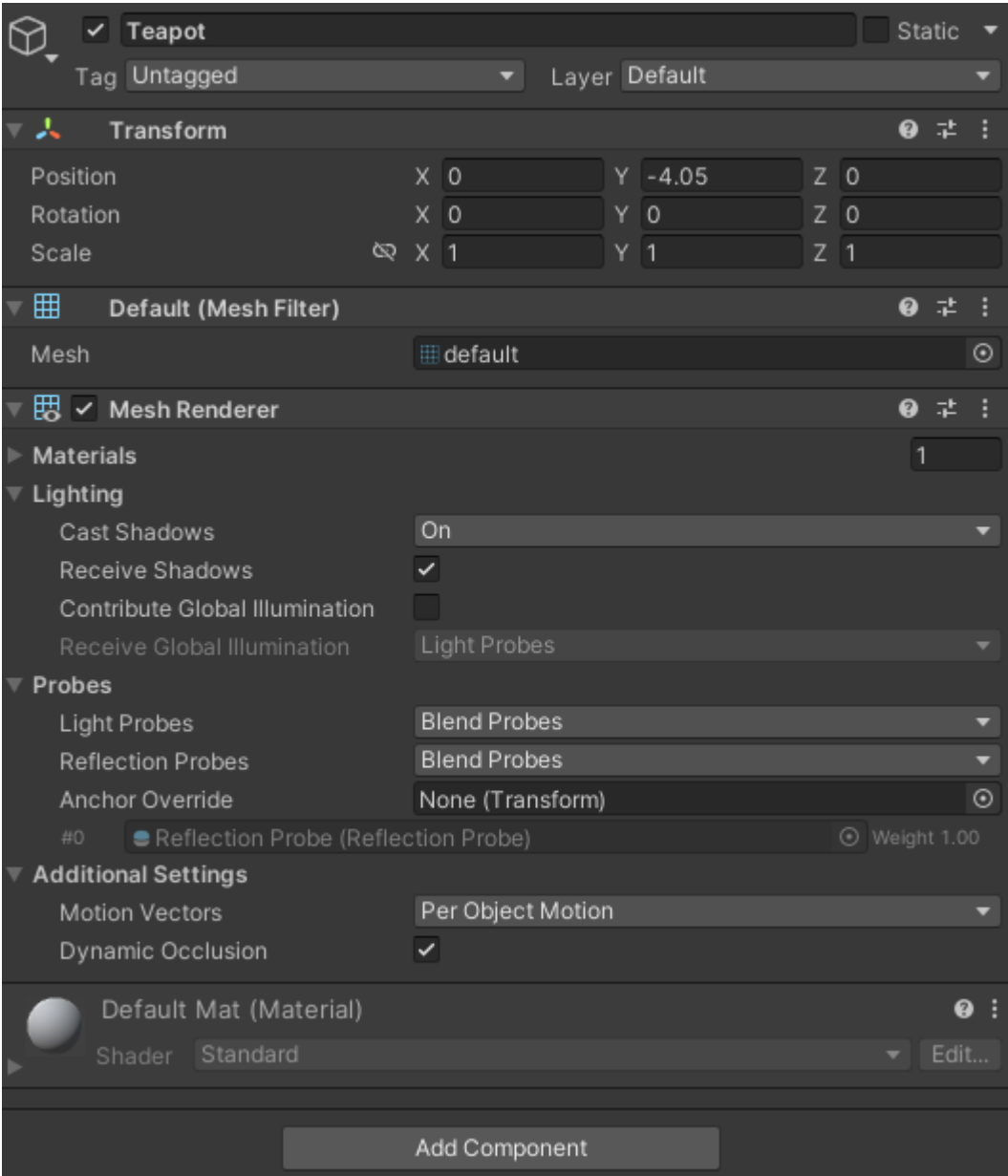
Adding Voxel Colliders

Before you can use the Voxel Colliders functionality you need to register the plugin:

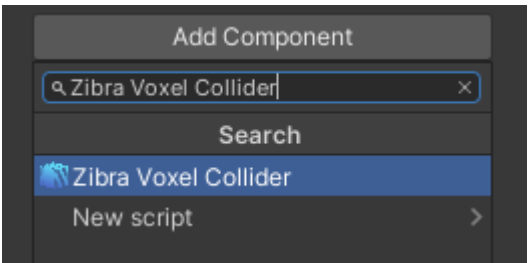
[Registering Zibra Liquids](#)

To add a voxel collider you first need to have the mesh that you want the water to collide with. In this example we'll use a teapot as an example.

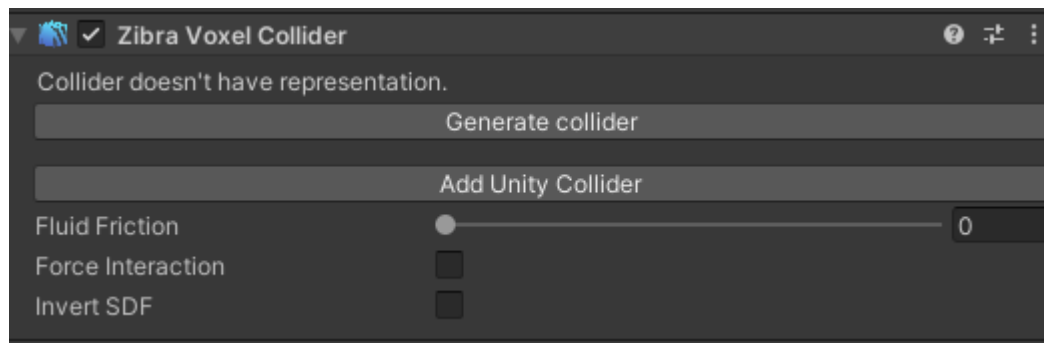
1. Open your GameObject with MeshFilter in Inspector



2. Press “Add Component” and select “Zibra Voxel Collider”

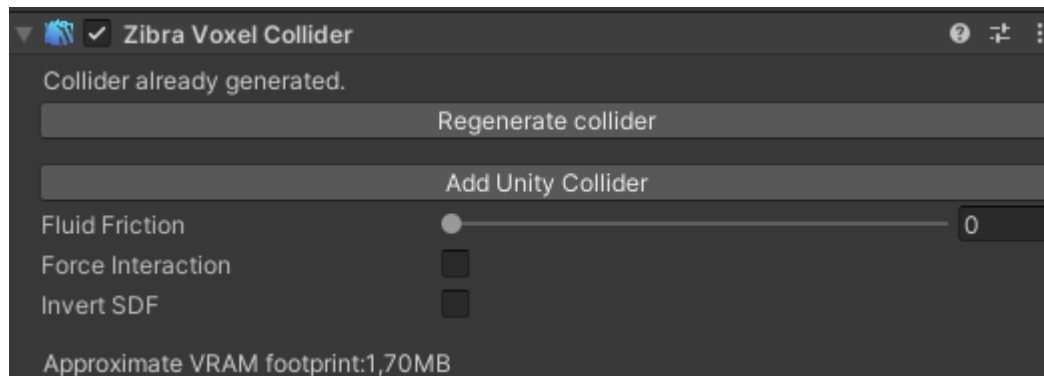


3. You should see these buttons:

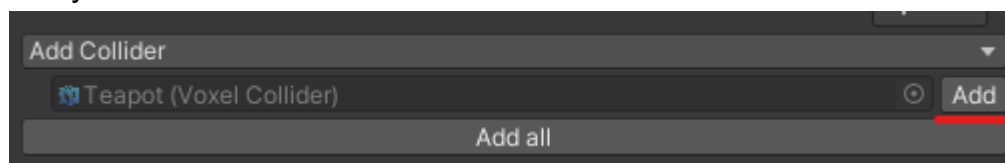


If you don't see them, you probably haven't registered your plugin yet. Register your plugin: [Registering Zibra Liquids](#) and try again.

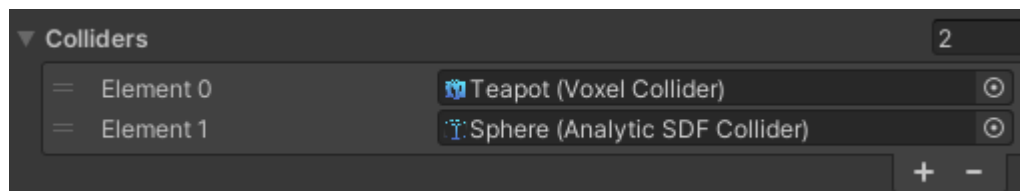
4. Press Generate Collider. **This will send your mesh to ZibraAI servers for processing.** This will generate neural representation for your mesh. After the generation has finished you'll see a message "Collider already generated", as well as the VRAM footprint of the collider.



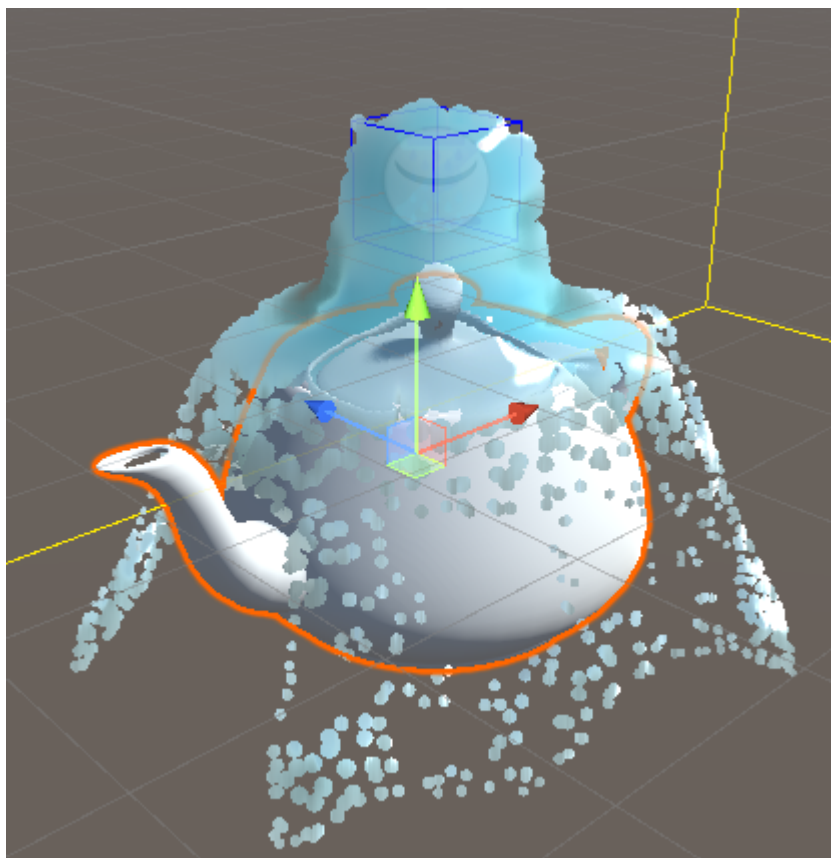
5. Open the Inspector window for Zibra Liquid and Press the Add button for your newly created collider.



6. You can now expand list of Colliders and see your colliders there



Now your collider is ready to be used. You can set its transform in the editor, or move it at runtime.



Collider parameters

- **Fluid Friction**

Fluid Friction 0

Controls the surface friction of the collider, used for liquid interaction.

- **Force Interaction**

Force Interaction ☐

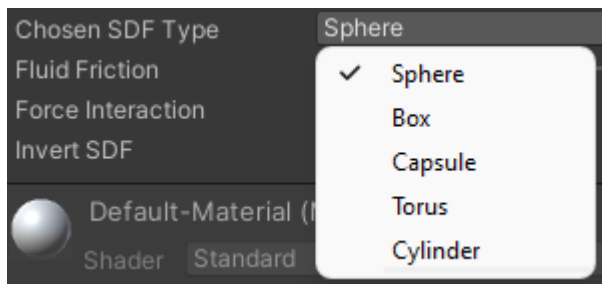
Controls whether liquid can apply force back to the object. You need to add a rigidbody component to the object that has a collider for liquid to apply force to.

- **Inverse SDF**

Invert SDF ☐

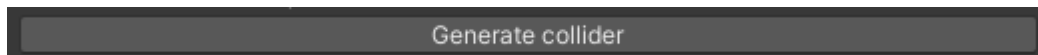
“Inverts” the collider. E.g. in the case of a sphere, liquid will only be allowed inside of that sphere.

- **Chosen SDF Type (*Analytic only*)**



Select the shape of the Analytic collider.

- **Generate Collider/Regenerate Collider (*button, Voxel only*)**



Generates a new, or regenerates an existing Voxel collider. You'll need to manually regenerate the collider if you change the mesh.

- **Add Unity Collider (*button, Voxel only*)**



Adds a Mesh collider, for the GameObject to collide with other meshes. Not required if you don't want liquid colliders to collide with non liquid objects. Only shown if you don't already have a Mesh collider.

Manipulators

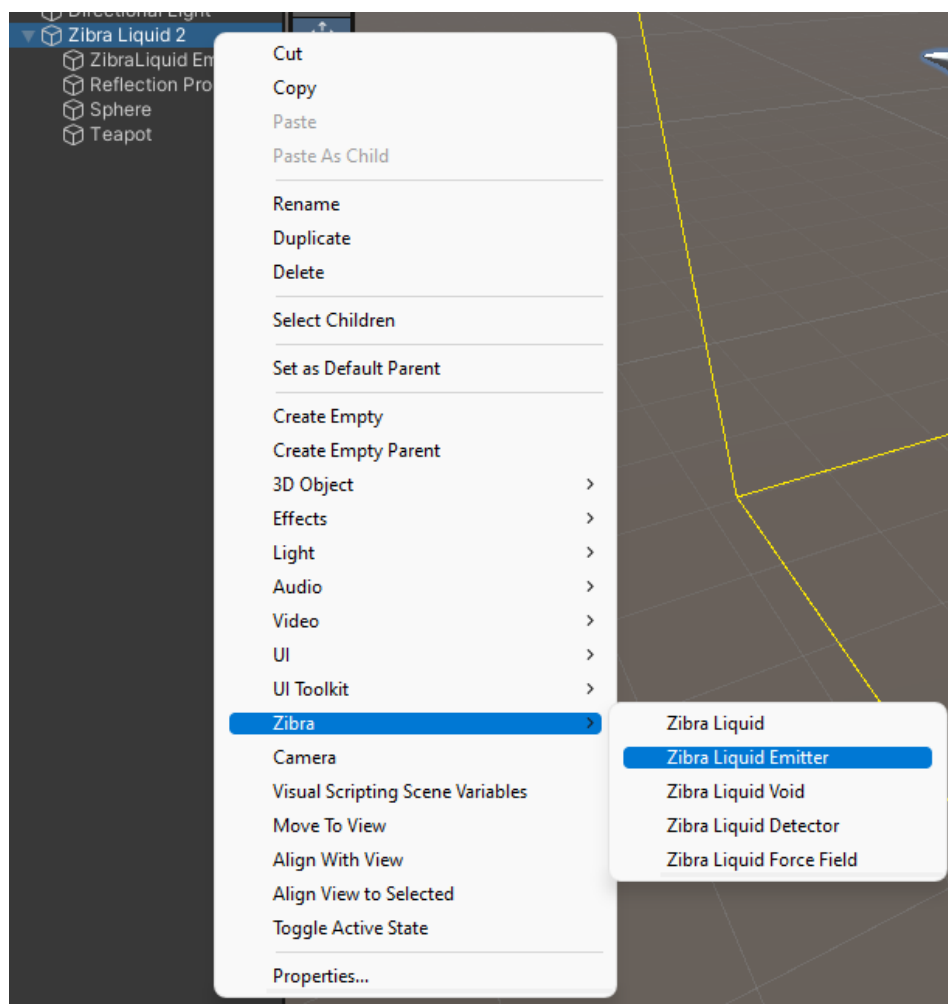
Zibra Liquid has many kinds of manipulators: Emitter, Void, Detector, Force Field

Emitter

Liquid emitters are used for emitting liquid, if you want to spawn liquid you need these. When you first create Zibra Liquid, the Emitter is automatically created and added to it. If you don't need additional ones you can skip adding additional emitters and go right to [Emitter parameters](#)

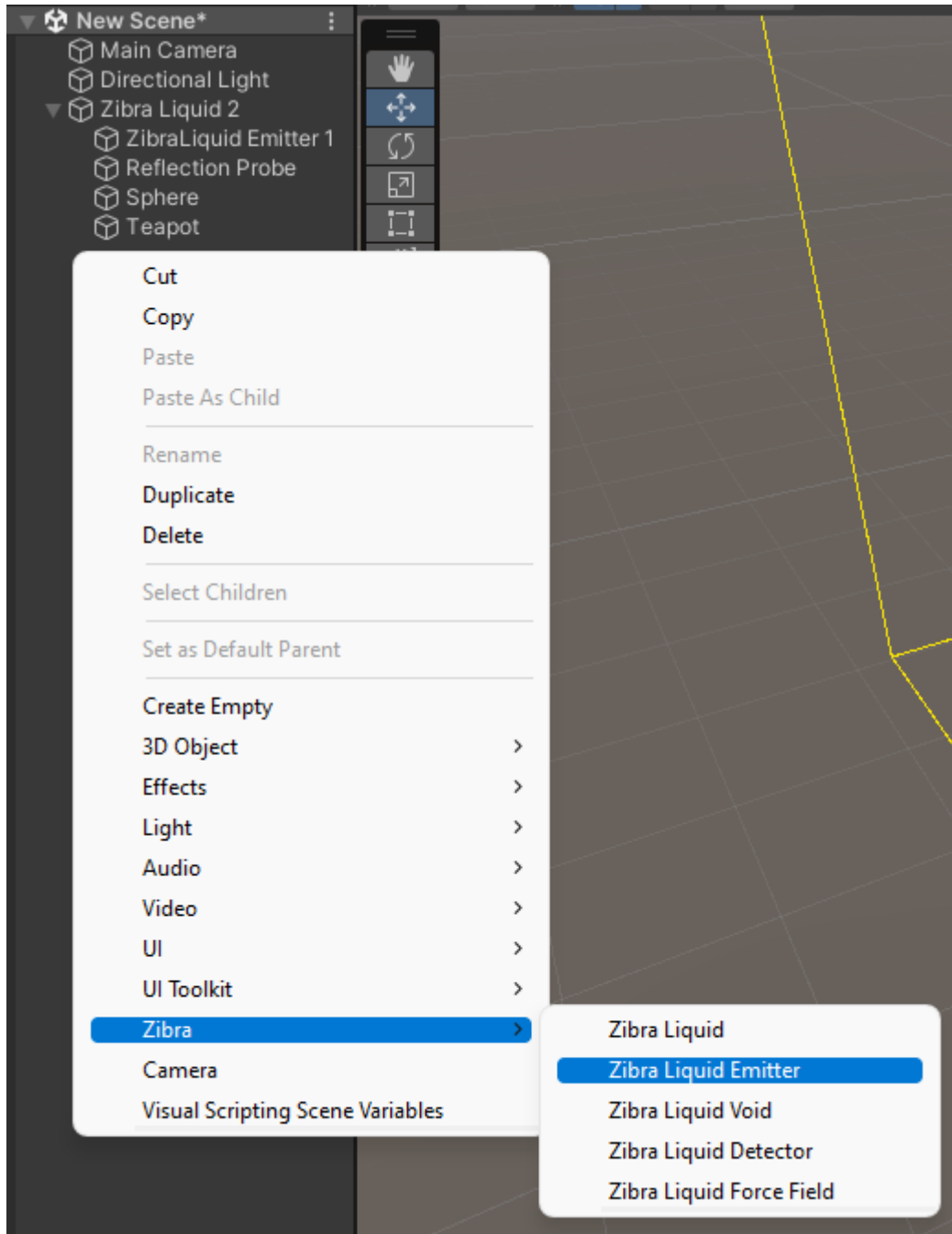
Adding Emitter

1. Right click on your Zibra Liquid instance, and select Zibra -> Zibra Liquid Emitter

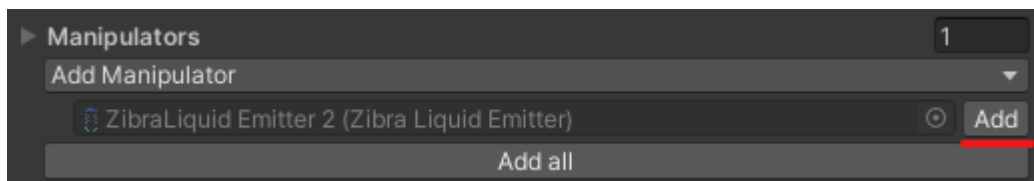


That will automatically add a new emitter to your liquid.
Alternatively, you can:

1. Right click in Hierarchy (*not on the Zibra Liquid instance*) and select Zibra -> Zibra Liquid Emitter



2. Open your Zibra Liquid instance in Inspector, and press the “Add” button for your newly created emitter under Manipulators list.



Emitter parameters

- **Emitter statistics (*readonly*)**

Total amount of created particles: 0
Amount of created particles per frame: 0

When you start the play mode you'll be able to keep track of how many particles were emitted. You can also access those values via scripts, via *"createdParticlesTotal"* and *"createdParticlesPerFrame"* attributes.

- **Particles per second**

Particles Per Sec 6000

Amount of liquid that is emitted per second.

- **Initial Velocity**

Initial Velocity X 0 Y 0 Z 0

Initial speed of the generated particles.

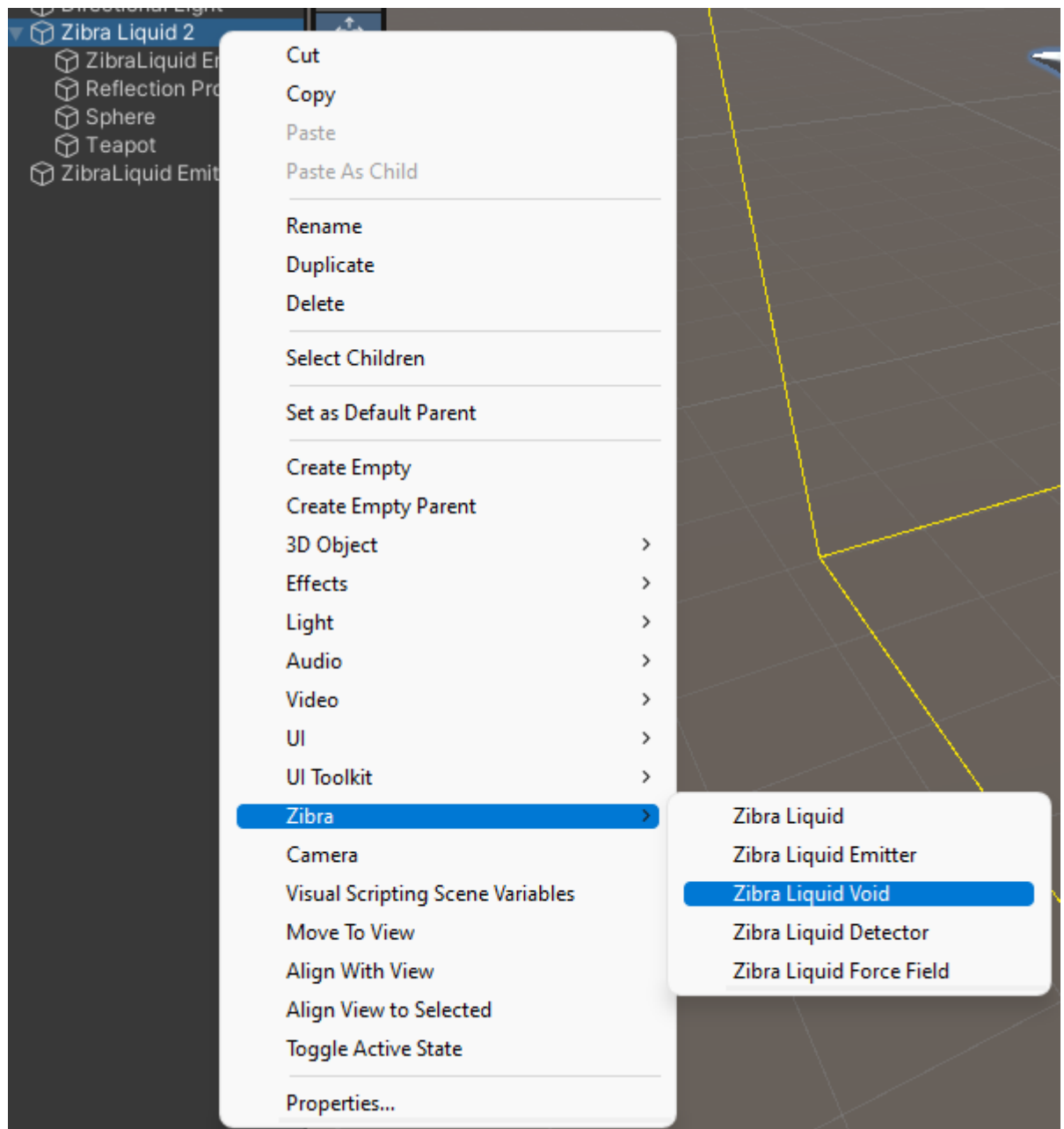
- Emitter's transform changed via changing GameObject's transform.

Void

Voids are used to destroy liquid. Since there's a limit to the maximum number of particles in Zibra Liquid instance, you need to destroy particles for your emitters to keep emitting liquid continuously. But if it's fine for you that the emitters will stop at some point, the usage of voids is optional.

Adding Void

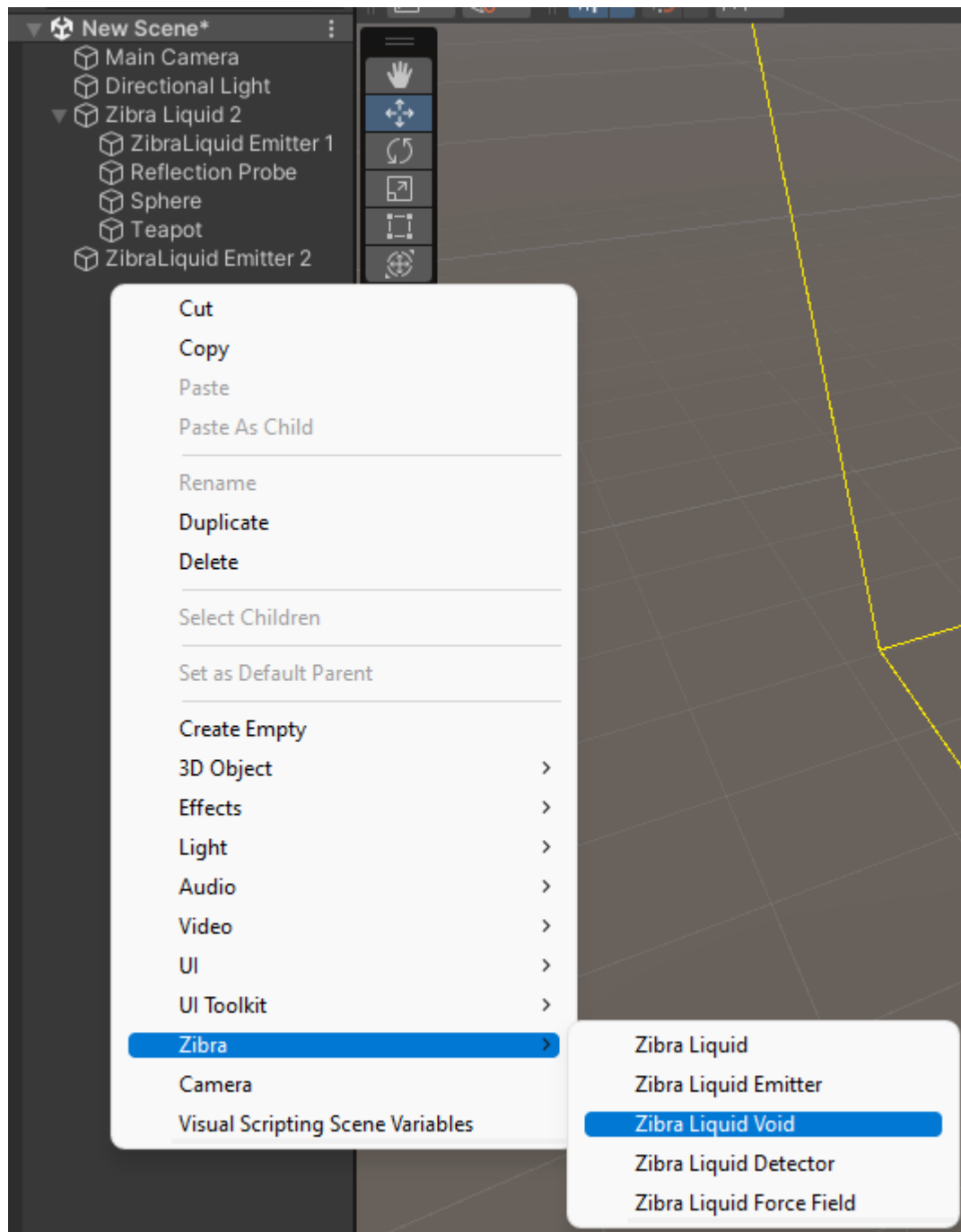
1. Right click on your Zibra Liquid instance, and select *Zibra -> Zibra Liquid Void*



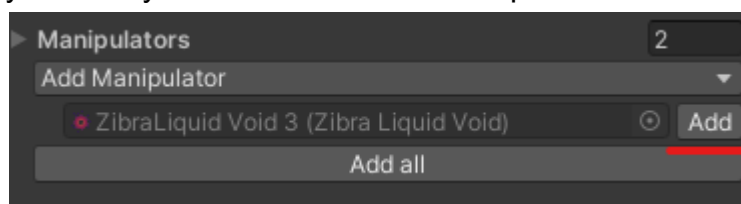
That will automatically add a new void to your liquid.

Alternatively, you can:

1. Right click in Hierarchy (*not on the Zibra Liquid instance*) and select *Zibra -> Zibra Liquid Void*.



2. Open your Zibra Liquid instance in Inspector, and press the “Add” button for your newly created void under Manipulators list.



Void parameters

- **Void statistics (readonly)**

Total amount of deleted particles: 0
Deleted particles per frame: 0

When you start play mode you'll be able to keep track of how many particles were deleted. You can also access those values via scripts, via `"deletedParticleCountTotal"` and `"deletedParticleCountPerFrame"` attributes.

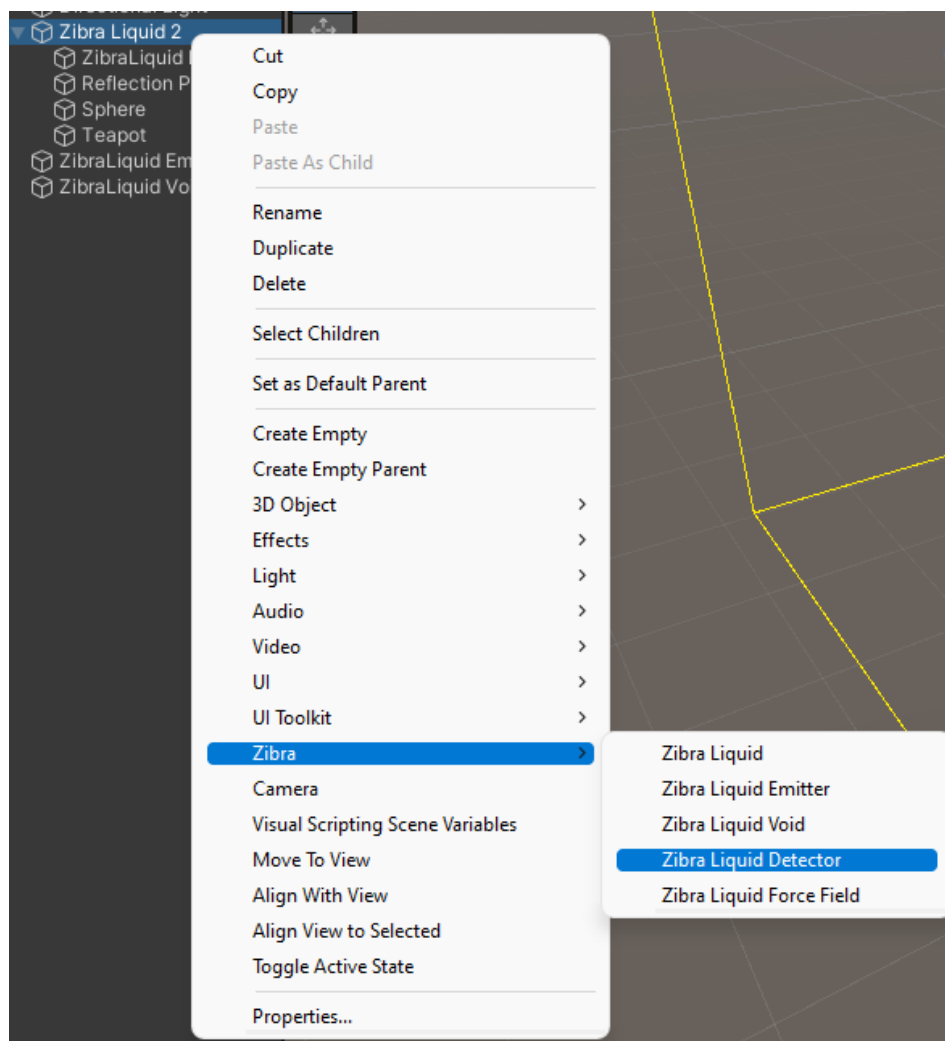
- Void's transform changed via changing GameObject's transform

Detector

Detector is used to detect how many liquid particles are in a specified volume.

Adding Detector

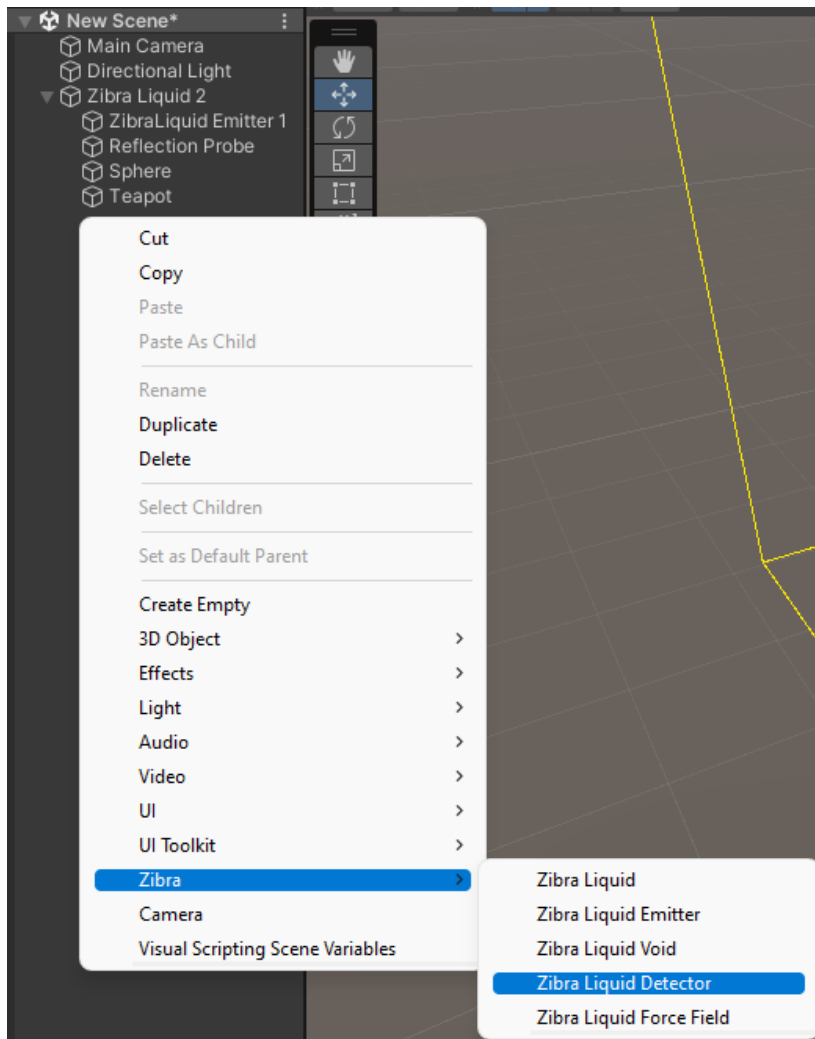
1. Right click on your Zibra Liquid instance, and select Zibra -> Zibra Liquid Detector



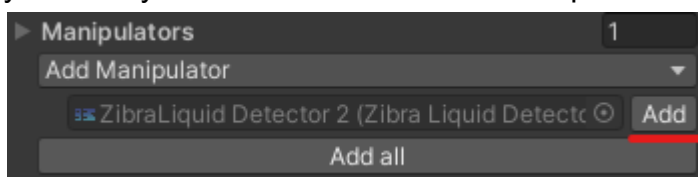
That will automatically add a new detector to your liquid.

Alternatively, you can:

1. Right click in Hierarchy (*not on Zibra Liquid instance*) and select *Zibra -> Zibra Liquid Detector*



2. Open your Zibra Liquid instance in Inspector, and press the “Add” button for your newly created Detector under Manipulators list.



Detector parameters

- Detected particle count (readonly)

Amount of particles inside the detector: 0

When you start play mode you'll be able to keep track of how many particles are inside the detector. You can also access those values via scripts, via the “particlesInside” attribute.

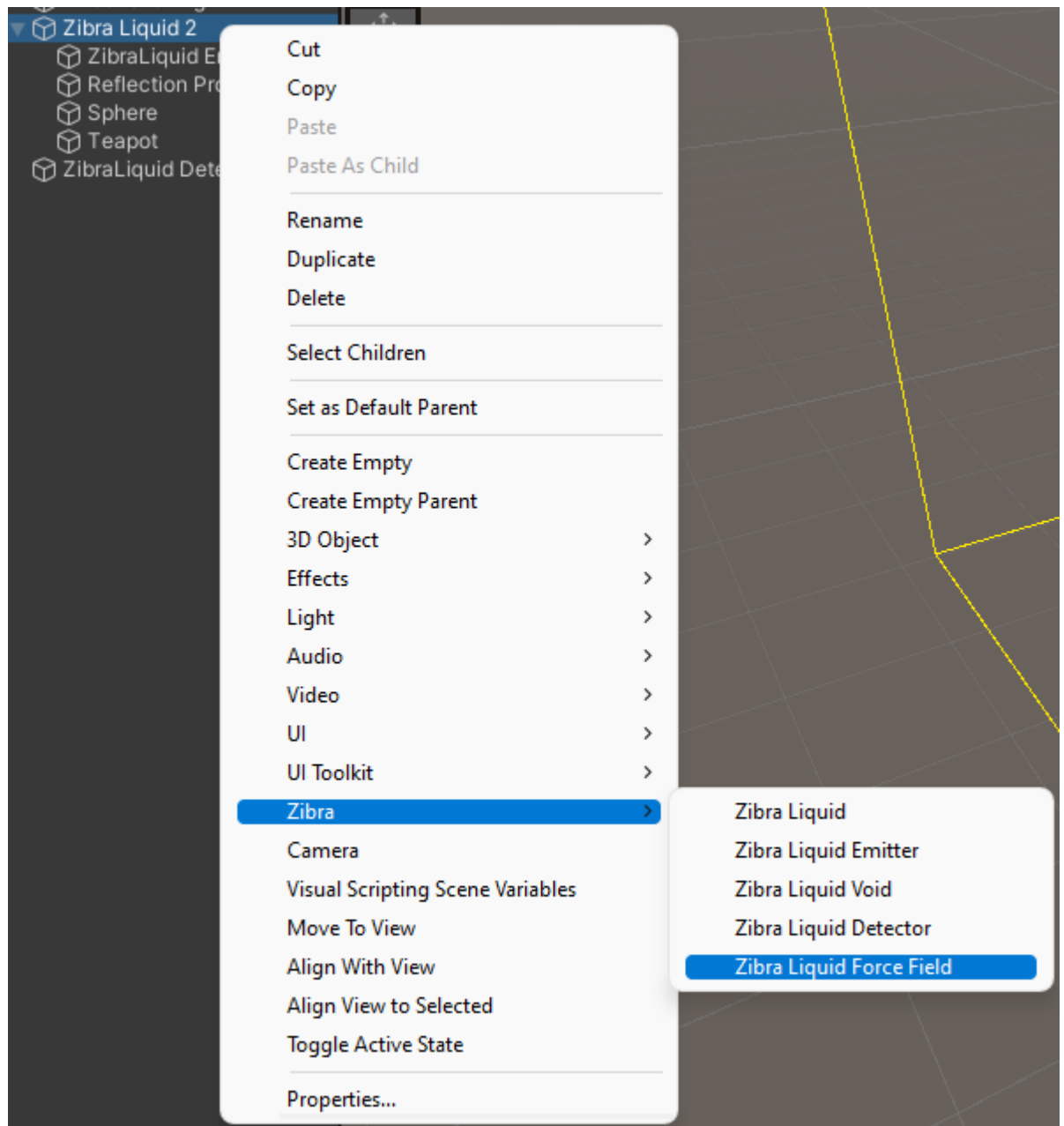
- Detectors' transform changed via changing GameObject's transform.

Force Field

Force fields are used to apply force to liquid in a specific way, in the specific region. You can for example use force fields to create radial gravity, to use instead of the default vector one. Or you can create a fountain by putting a force field that pushes liquid up in the body of water.

Adding a Force Field

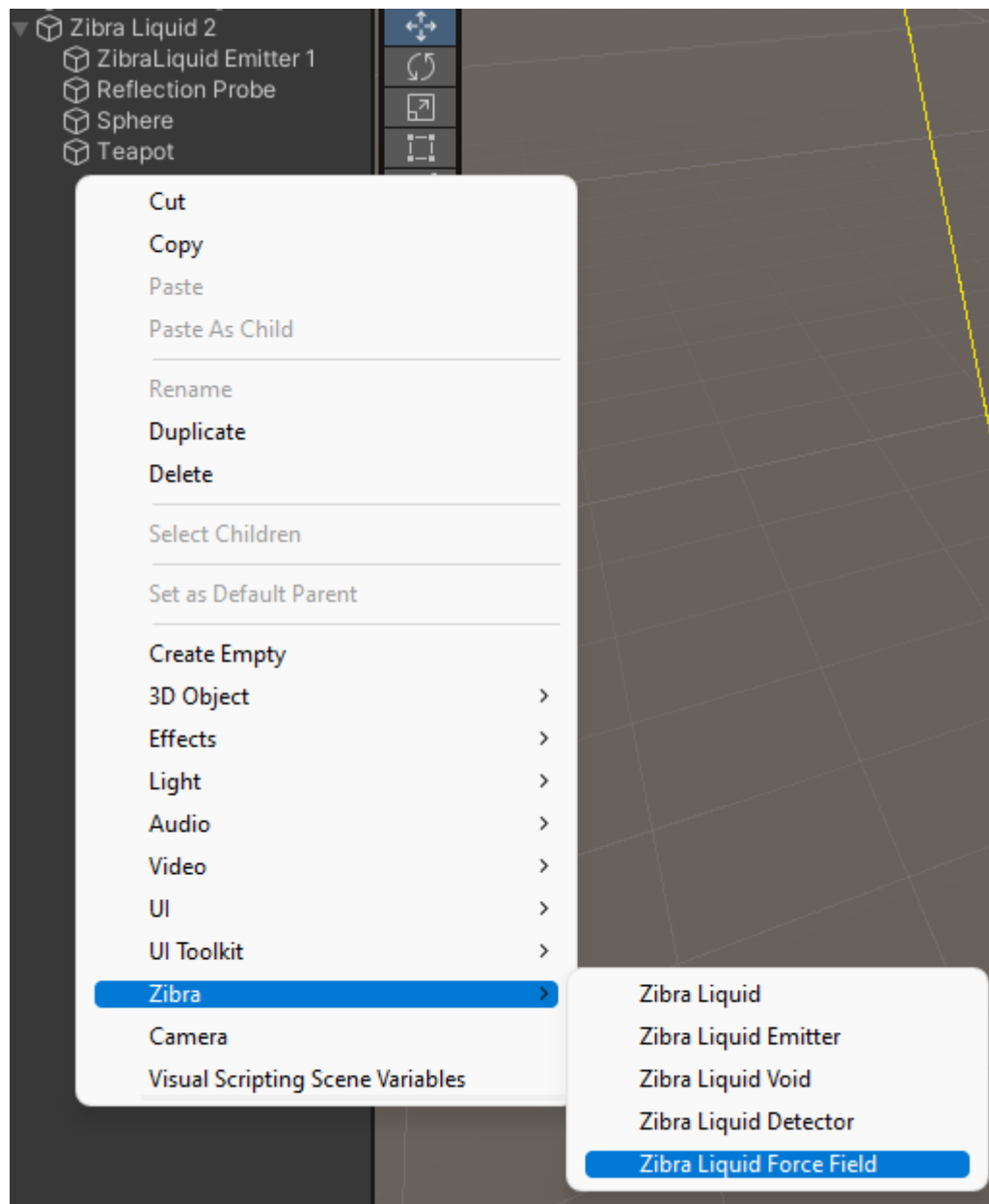
1. Right click on your Zibra Liquid instance, and select Zibra -> Zibra Liquid Force Field



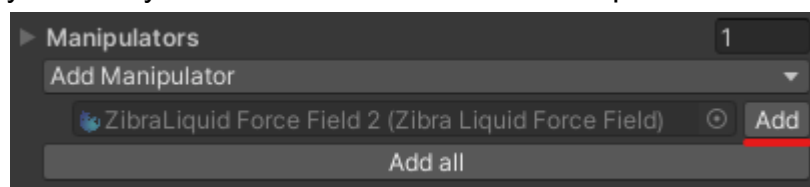
That will automatically add a new detector to your liquid.

Alternatively, you can:

1. Right click in Hierarchy (not on Zibra Liquid instance) and select Zibra -> Zibra Liquid Force Field

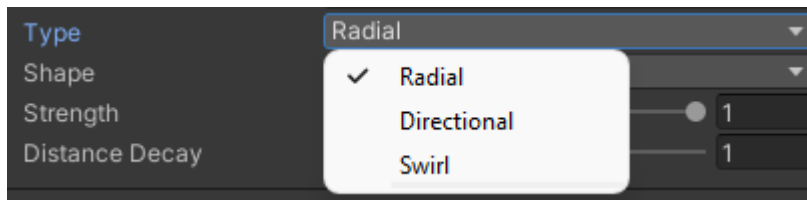


2. Open your Zibra Liquid instance in Inspector, and press the “Add” button for your newly created Force Field under Manipulators list.



Force Field parameters

- **Type**



Configures how Force Field applies force to liquid.

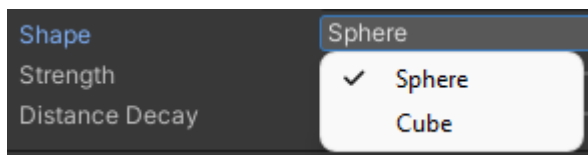
Radial - pushes or pulls liquid towards the Force Field.

Directional - pushes liquid in a specified direction.

Swirl - creates a whirlpool force around the Force Field.

In case of Directional and Radial settings, direction can be adjusted by rotating the Force Field.

- **Shape**



Configures which shape Force Field will have. Important for the distance between the particle and Force Field calculation.

- **Strength**



Configures how strong the strength of the force applied by Force Field will be.

Can be negative to reverse direction. (For radial type, positive force is pulling, and negative is pushing)

- **Distance decay**



Configures how fast the force strength decays with the distance from the Force Field.

Liquid Initial State Baking (experimental)

If you want to have Zibra Liquid instance pre-filled on startup you need to use Initial State baking.

After baking, the Zibra Liquid instance will remember its initial state and will start simulation from it. Note that we do not support baking animation, simulation over time, etc. Simulation itself is always performed on device in real time, only the initial state is baked.

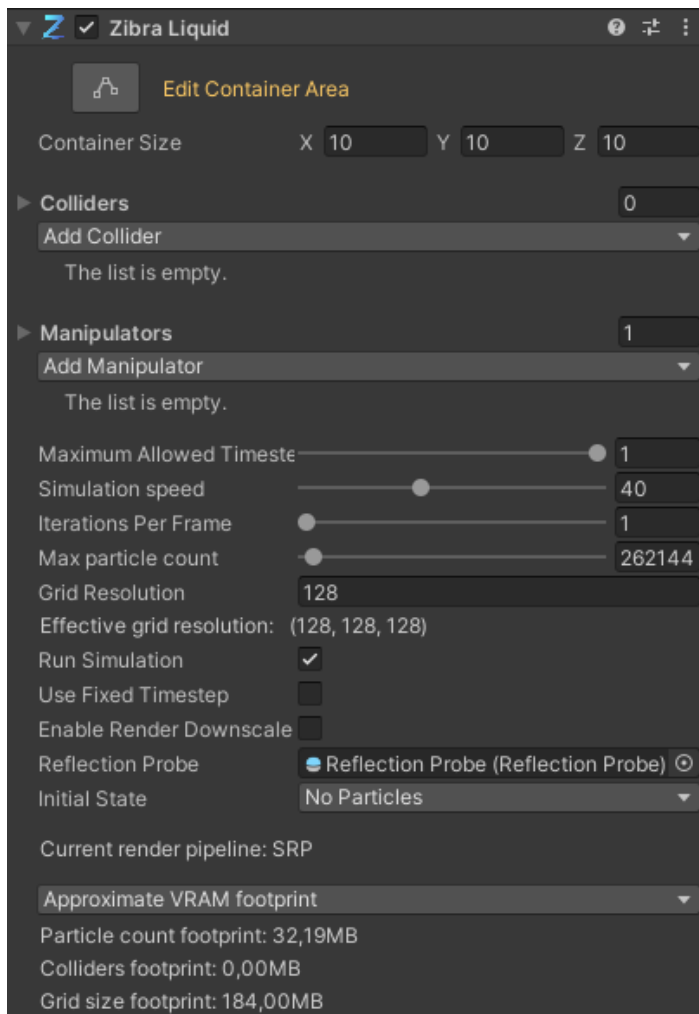
Initial State Baking will simulate the first N seconds of your liquid instance, save it, and simulation in play mode will start from that point.

Note that after changing some parameters you'll have to rebake your liquid, specifically: Grid Resolution, Container Size (in case you changed the size non-uniformly, and as a result changed the Effective grid resolution), Max particle count (in case your baked state has more particles than the current max number).

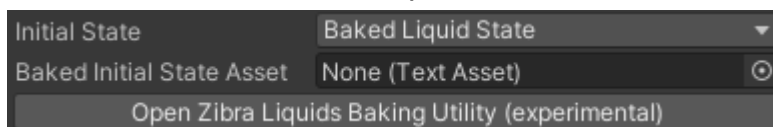
To bake initial state:

1. If you haven't already, add all emitters, voids, and colliders that you are going to have to your liquid.
2. Open the baking utility and select your Zibra Liquid instance to bake.
There are 2 options to do it:
 - a. Via Zibra Liquid's Inspector

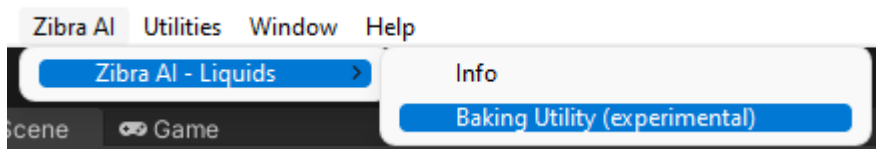
- i. Open Zibra Liquid's Inspector



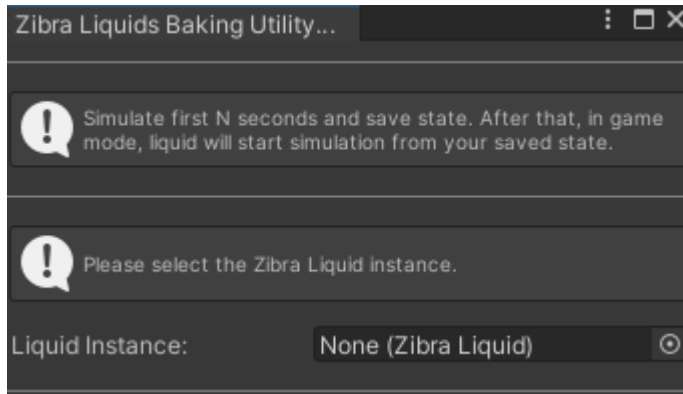
- ii. Set Initial State to “Baked Liquid State”. New button will appear.



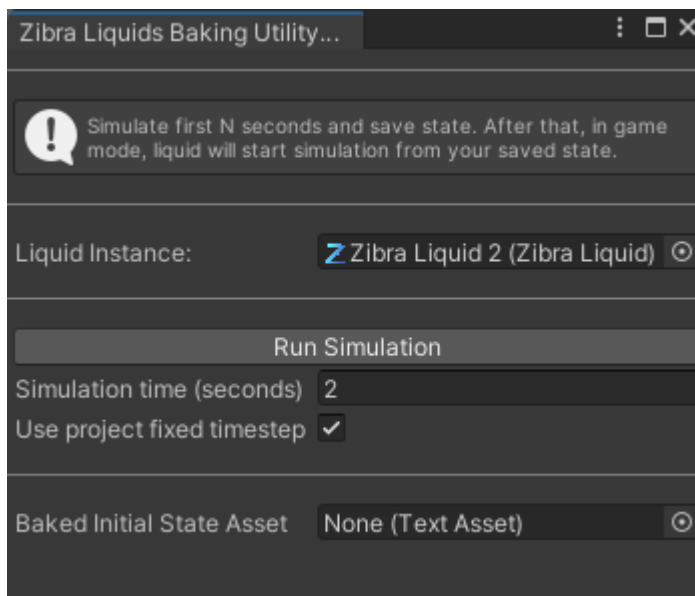
- iii. Press “*Open Zibra Liquids Baking Utility (experimental)*”
 - b. Via Unity’s menu
- i. Press “*Zibra AI -> Zibra AI - Liquids -> Baking Utility (experimental)*”



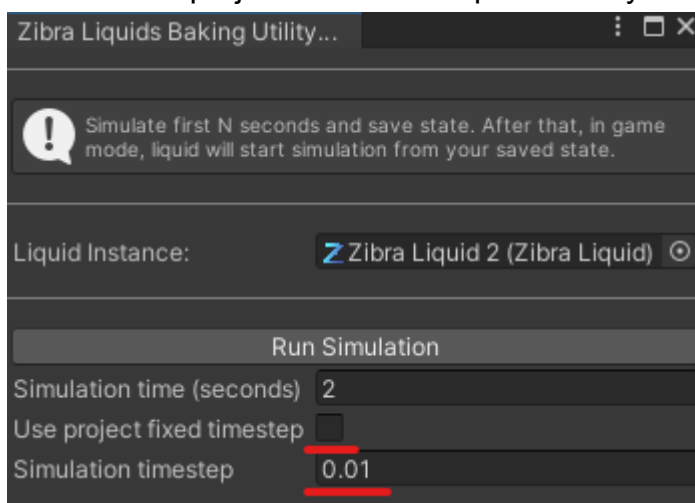
- ii. If the Zibra Liquid instance wasn't selected automatically, select the Liquid Instance you want to bake.



3. Now you see Baking Utility window

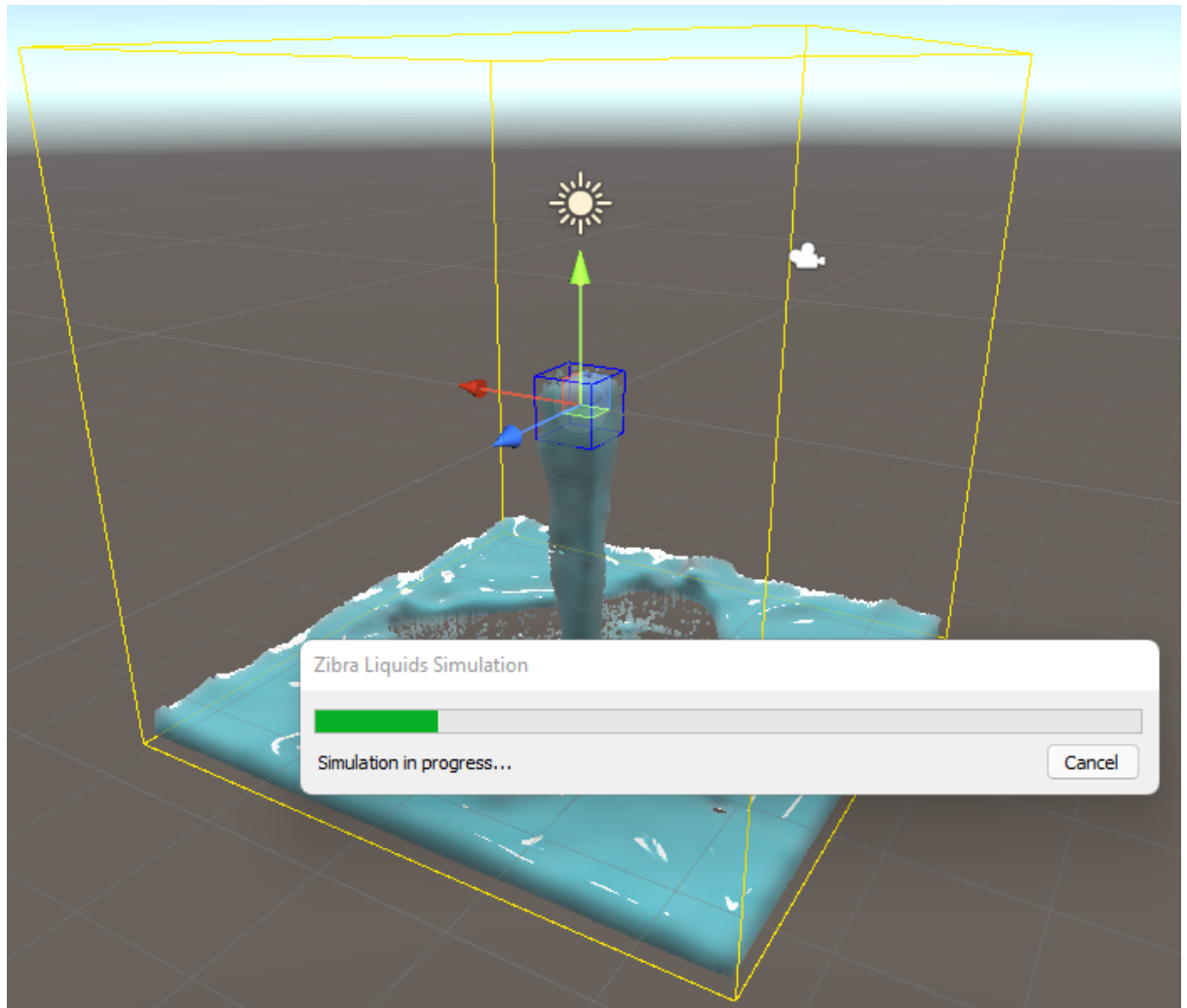


4. Set the Simulation time to the number of seconds you want to simulate.
5. (optional) If you want to run baking simulation with specific timestamp you can disable "Use project fixed timestep" and set your desired "Simulation timestep"

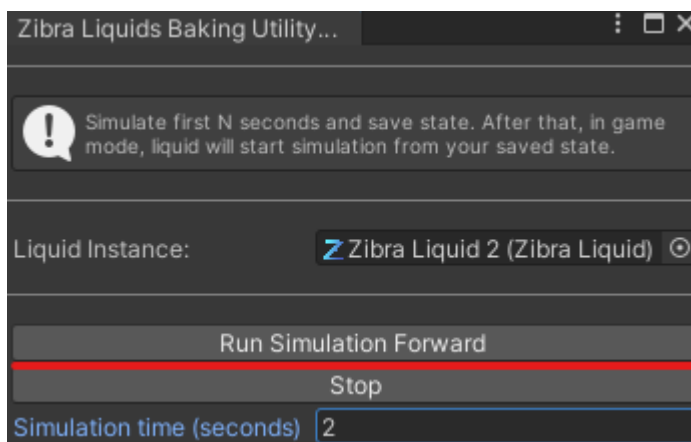


6. Press *“Run Simulation”*.

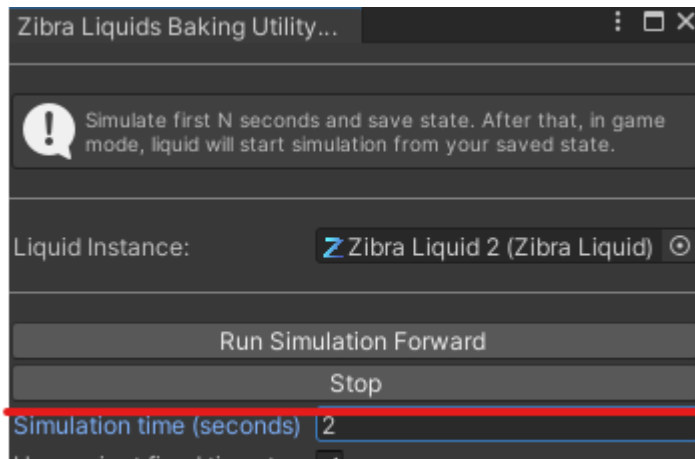
You'll see simulation running in Scene/Game view (*only Game view on URP*)



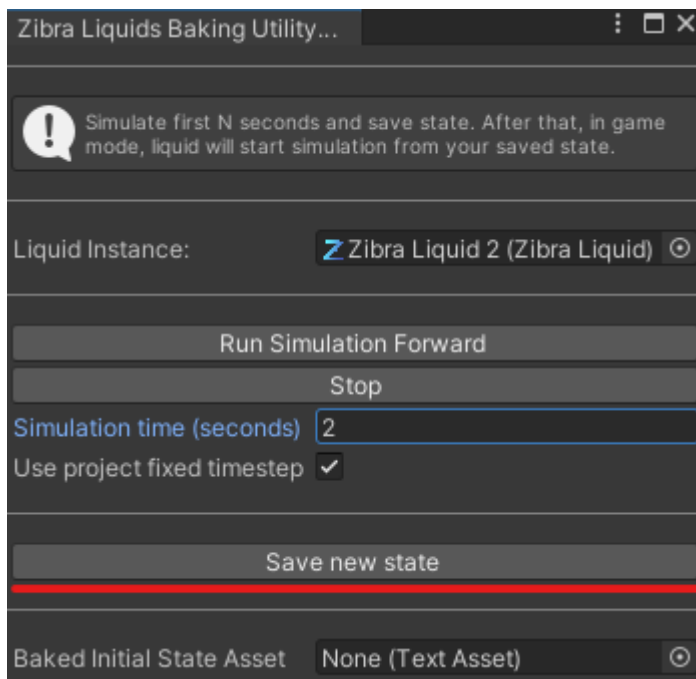
7. (*optional*) If you want to run simulation further, set Simulation time to the amount of time you want to run simulation forward, and press *“Run Simulation Forward”*.



- (optional) If you simulated too far, and want to start over, press “Stop” and go back to step 4

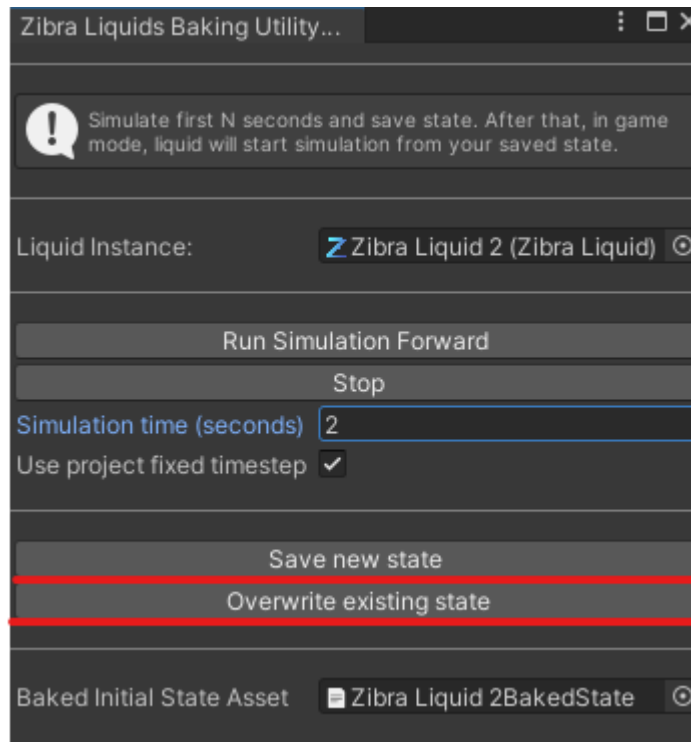


- Press “Save new state” to save baked state.



That will save the new state to the file in the subdirectory next to your scene file. e.g. for Liquid Instance “ZibraLiquid1” in “/Assets/NewScene.unity” scene, liquid baked state will be saved into “/Assets/NewScene/” folder as “ZibraLiquid1BakedState.bytes”.

- a. If you already had a previous saved state you'll have 2 buttons: "Save new state" and "Overwrite existing state".



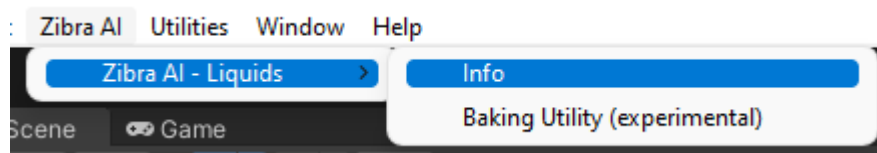
- b. "Save new state" always creates new file, while "Overwrite existing state" always overwrites the old file with a new state.
10. (optional) After you are done with baking, press "Stop" to stop the liquid simulation and unlock the parameters that are disabled when liquid is "live". Next time you'll change scene or enter playmode this state will reset, so you usually don't need to manually stop it.

Registering Zibra Liquids

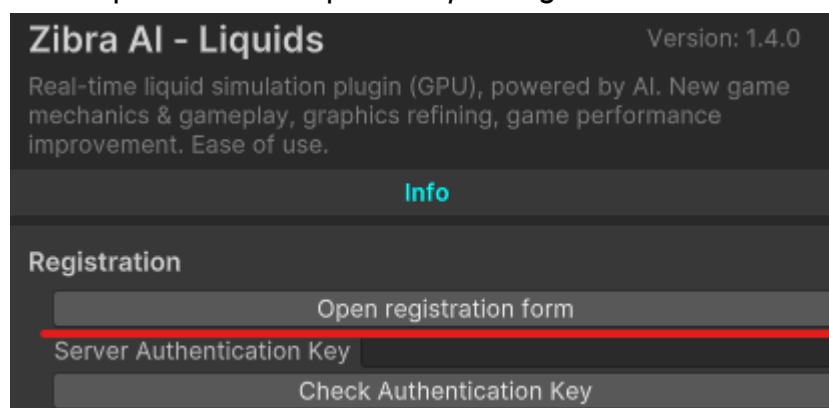
Access to Voxel Collider generation requires registering, since that feature requires access to Zibra AI's server for processing.

To register your copy of Zibra Liquids:

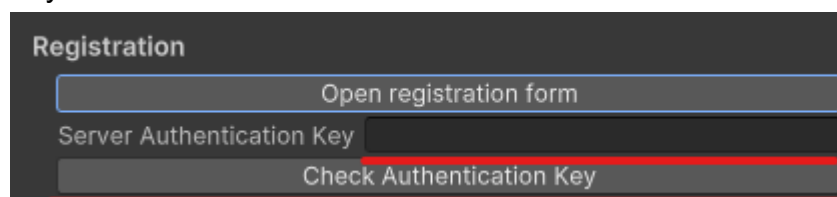
1. Make sure that you launched Unity Editor via Unity Hub and are logged into the correct Unity account.
2. Press "Zibra AI -> Zibra AI - Liquids -> Info"



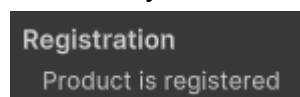
3. In the opened window press "Open registration form"



4. Fill out the registration form. Note that you'll need an Invoice number. It is not the same number as the order number. You can find your invoice number in your emails. It has the following format: "IN012345678901".
5. After filling out the form, wait for an email with a Server Authentication Key.
6. Enter key into "Server Authentication Key" field and press Check Authentication Key.



7. After that you should see this message



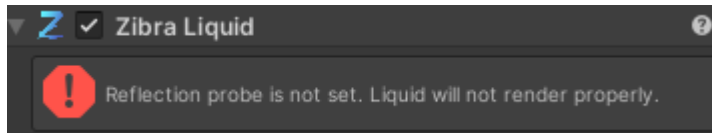
After registration you can generate voxel colliders. **Note that generating voxel colliders sends your mesh to the Zibra AI servers for processing.**

Troubleshooting

If you have any issues that you think are the fault of Zibra Liquids, please feel free to contact us on [Discord](#)

Some common issues you may have and ways to troubleshoot:

- Check whether you have any errors in Liquid Instance
e.g.



- On URP, not adding “URP Liquid Rendering Component”
See [Additional setup on URP](#)
- Trying to use Zibra Liquids on an unsupported platform (*Android, Oculus/Meta Quest 2*) or with unsupported features (*VR*)
See [System Requirements](#).
- On HDRP, Liquid is too bright
Decrease “*Reflection color*” in Liquid Instance’s Material Parameters
- Black artifacts on liquid near surface discontinuity
Decrease “*Blur radius*” in Liquid Instance